Macro Trends, Complexity, and the Current Status of Software Defined Networking

David Meyer
CTO and Chief Scientist, Brocade
Director, Advanced Technology Center, University of Oregon
NSF SDN Program Review
Arlington, VA
dmm@{brocade.com,uoregon.edu,1-4-5.net,...}
http://www.1-4-5.net/~dmm/talks/nsf_sdn_program_review.pdf

Agenda

- Too many words, too many slides ☺
  - This talk is about thinking about SDN (and networking) in new ways
    - Perhaps more questions than answers
  - I'm going to wind up skipping many of these slides…included for context
- (One) Macro Trend
- Context: SDN Problem Space and Hypothesis
  - And How We Got Here
- Complexity, SDN and Universal Principles
- SDN: Architecture and where we're going
- Summary and Q&A if we have time

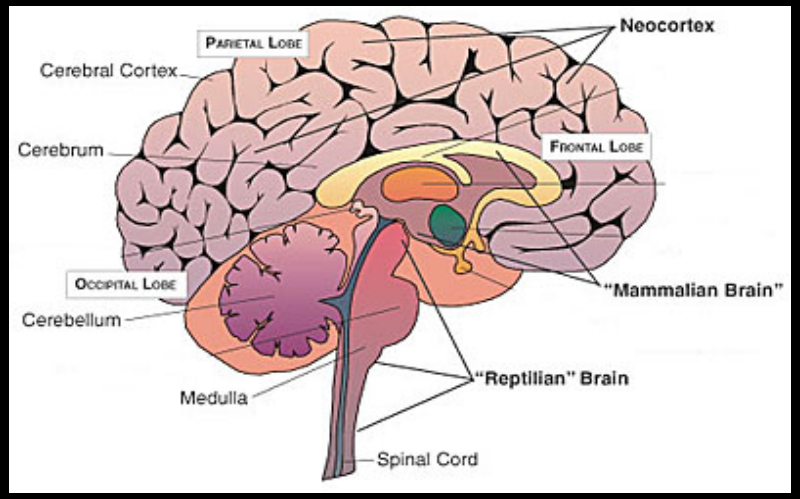
Danger Will Robinson!!!

This talk might be controversial/provocative (and perhaps a bit "sciencey")

Macro Trends

Trend: The Evolution of Intelligence

Precambrian (Reptilian) Brain to Neocortex → Hardware to Software

HARDWARE	SOFTWARE
	

- Key Architectural Features of Scalable/Evolvable Systems
  - **RYF-Complexity**
  - **Bowtie architectures**
  - **Layered with distributed (integral)l feedback control**
  - Protocol Based Architectures

Once you have the h/w

its all about code

Agenda

- ~~Macro Trends?~~
- Context: SDN Problem Space and Hypothesis
- Complexity, SDN and Universal Principles
- SDN: Architecture and where we're going
- Summary and Q&A if we have time

Oh Yeah, This Talk Was Supposed To Have Something To Do With SDN (aka -- How We Got Here)

- Well then, what *was* the SDN problem space?

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

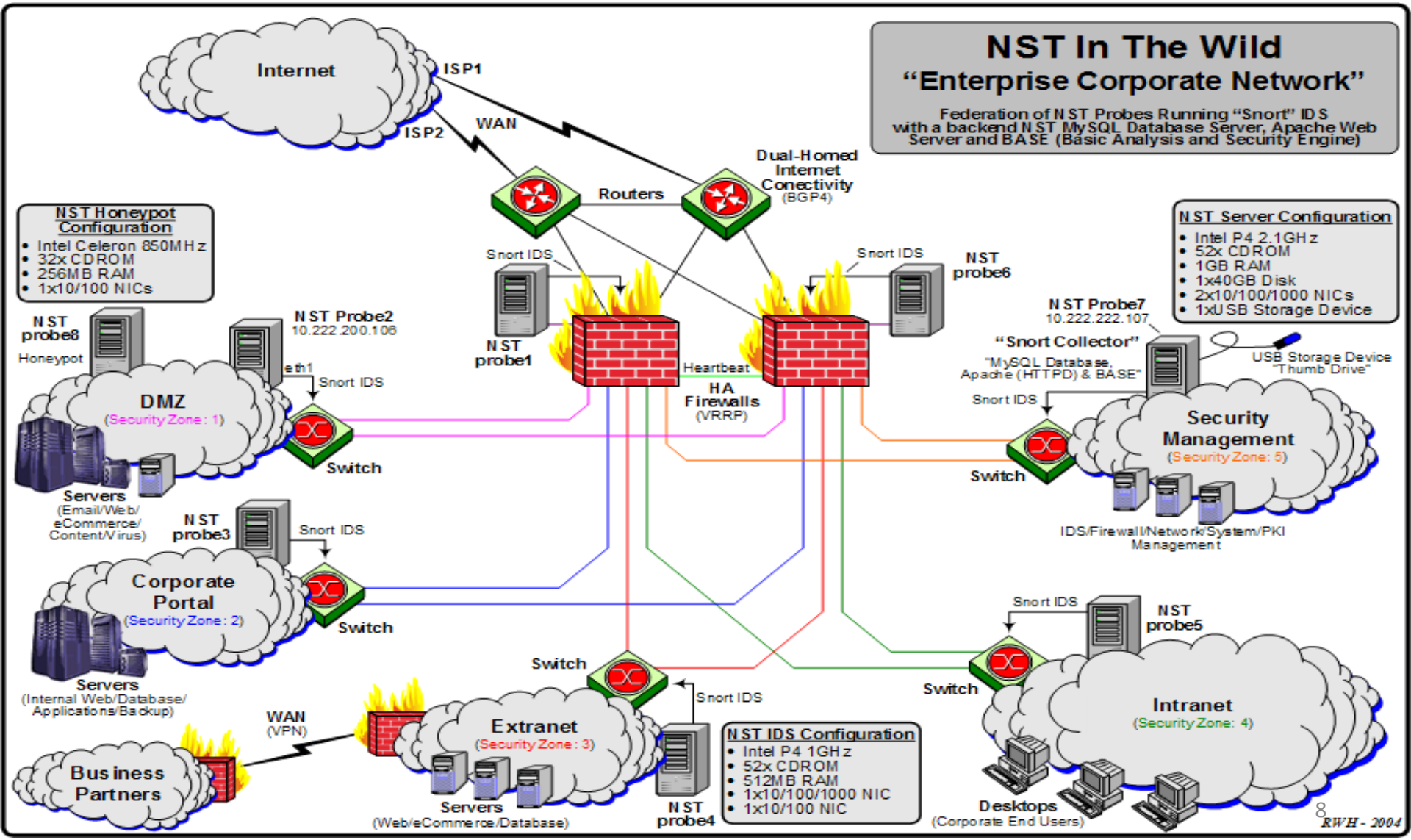
  - ***Provide state of the art evolvable network infrastructure and services while minimizing TCO***
  - → better, faster, cheaper, choose 3?

- ***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…

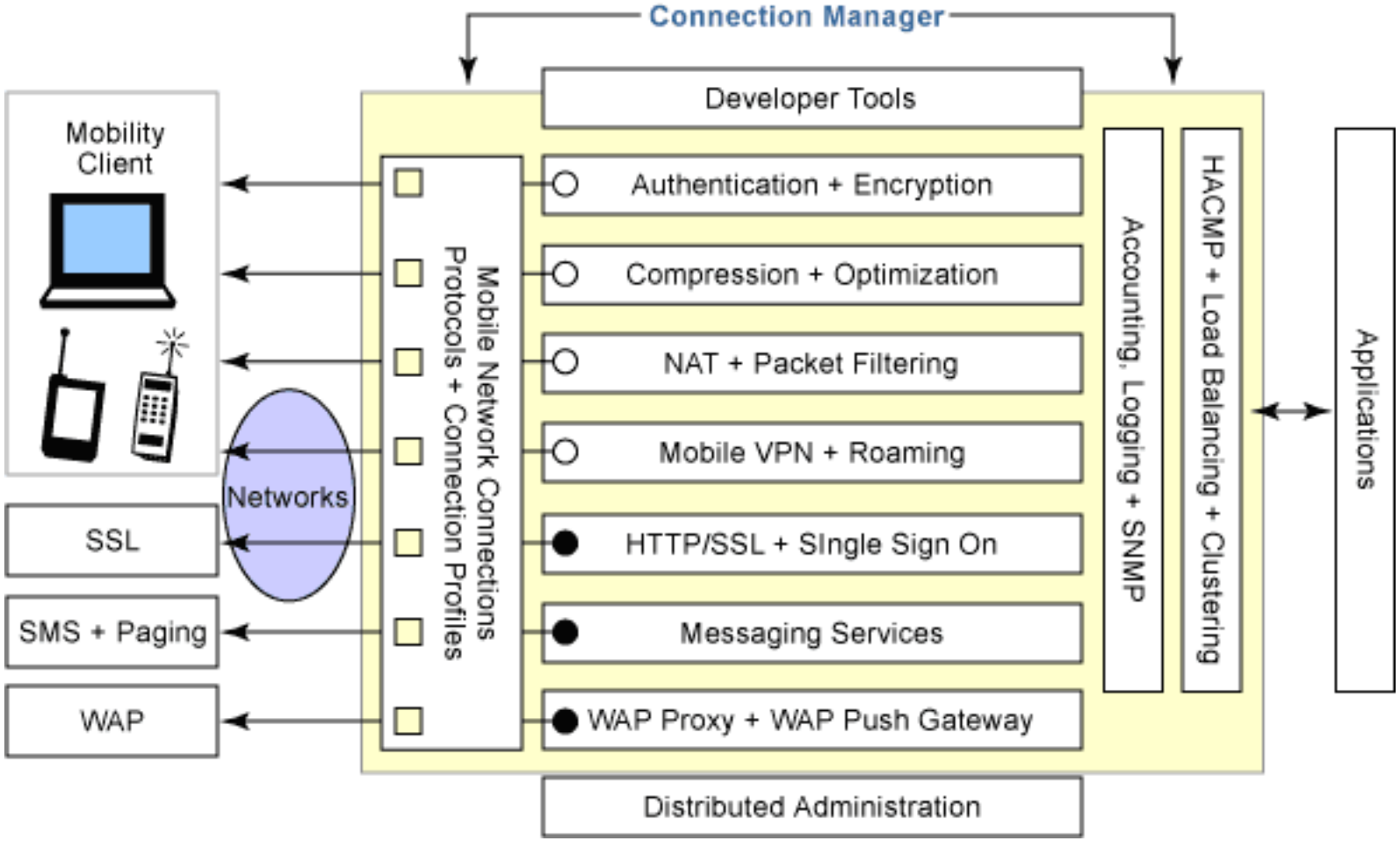
- *Note future uncertain*: Can't *"skate to where the puck is going to be"* because curve is unknowable (this is a consequence, as we will see, of the "software world" coupled with Moore's law and open-loop control).
  - That is, there is quite a bit of new research that suggests that such uncertainty 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 Robust *and* Fragile → The network is RYF-complex

Agenda

- ~~Macro Trends?~~
- ~~Context: SDN Problem Space and Hypothesis~~
- Complexity, SDN and Universal Principles
- SDN: Architecture and where we're going
- Summary and Q&A if we have time

Connecting Complexity, Design, and Robustness

"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.*"

See Alderson, D. and J. Doyle, "Contrasting Views of Complexity and Their Implications For Network-Centric Infrastructures", IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 40, NO. 4, JULY 2010

Robustness is a Generalized System Feature

- **Scalability** is robustness to changes to the size and complexity of a system as a whole
- **Evolvability** is robustness of lineages to large changes on various (usually long) time scales
- Other system features cast as robustness
  - **Reliability** is robustness to component failures
  - **Efficiency** is robustness to resource scarcity
  - **Modularity** is robustness to component rearrangements
- Not surprisingly, these are the same features we're seeking from the network

Just so we're all talking about the same things – a few definitions

- **Robustness** is the preservation of a certain property in the presence of uncertainty in components or the environment
  - Systems Biology: Biological systems are robust if their important functions are insensitive to the *naturally occurring variations* in their parameters
    - Limits the number of designs that can actually work in the real environment
    - Examples: Negative autoregulation and exact adaptation in bacterial chemotaxis

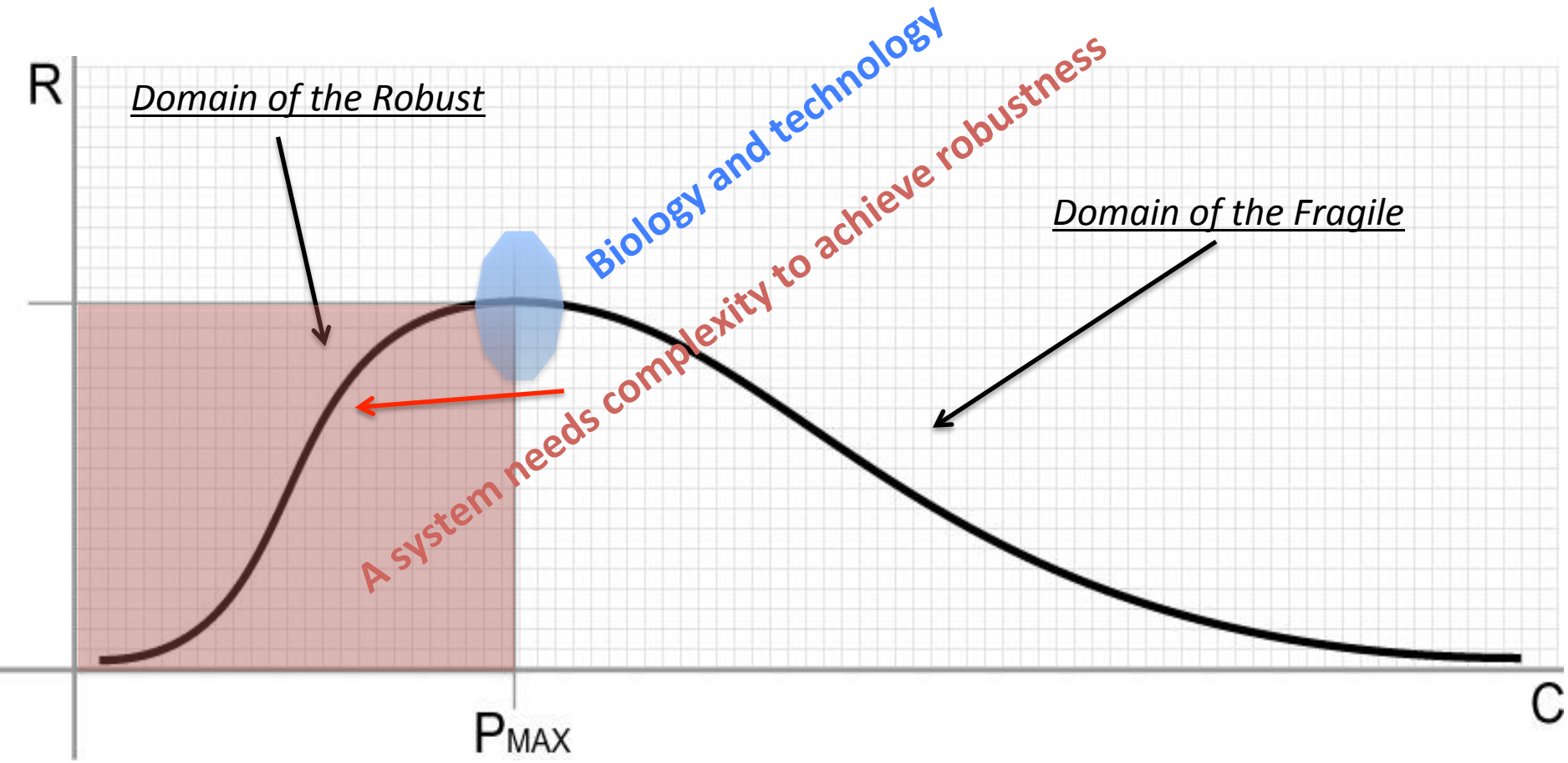
- **Fragility** is the opposite of robustness
  - Both need to be specified in terms of a system, a property and a set of perturbations

- 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***
  - Or the system may collapse if it experiences perturbations above a certain threshold (K-fragile)

- For 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
  - Another example: HSRP (VRRP) provides robustness to failure of a router/interface, but introduces fragilities in the protocol/implementation
  - Complexity/Robustness Spirals

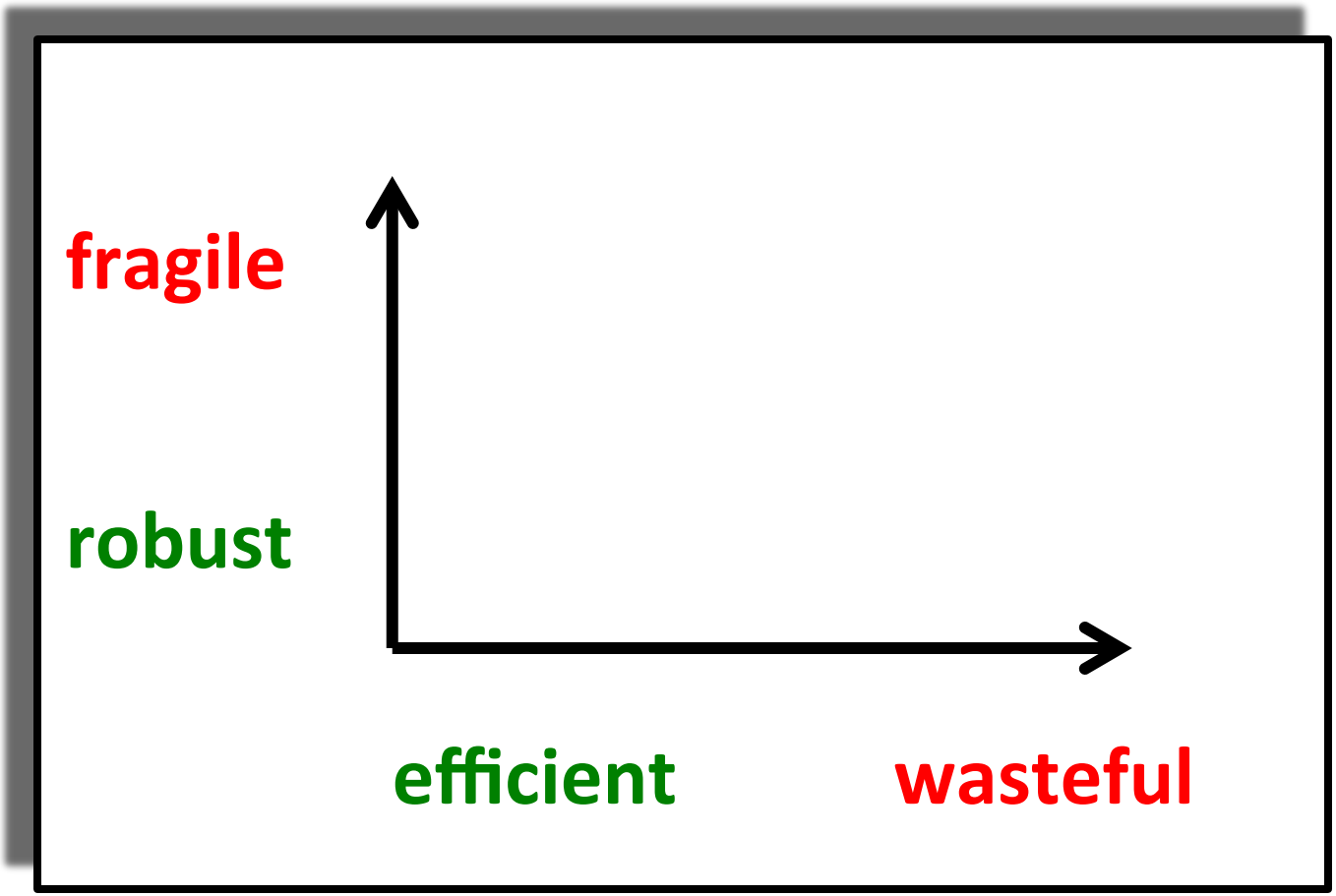
- **Takeaway**: Software, and SDN in particular, creates all kinds of RYF tradeoffs

BTW, Complexity Isn't Inherently "Bad"

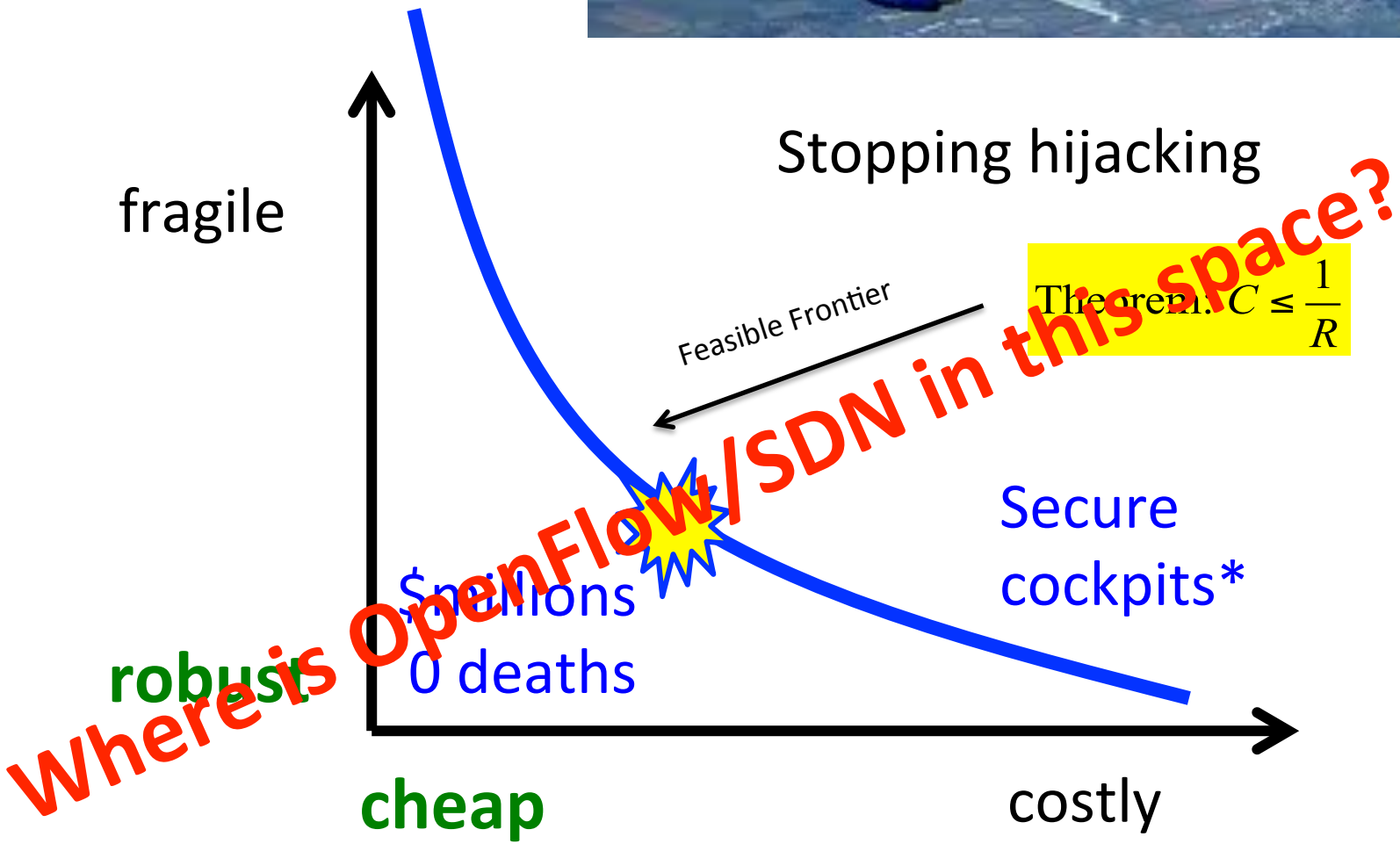


Increasing number of policies, protocols, configurations and interactions (well, and code)

Can we model the RYF tradeoff space as simple dichotomous pairs?

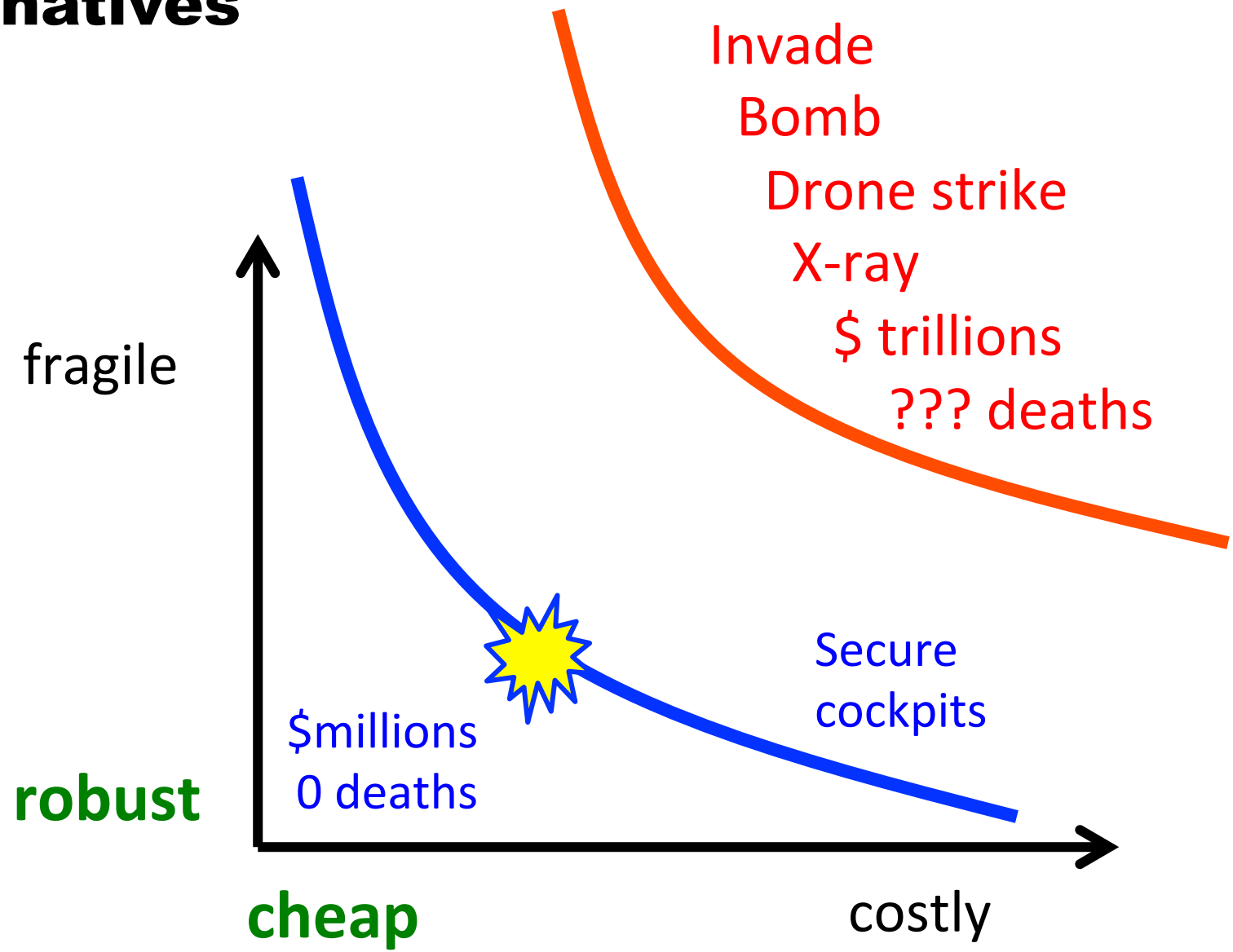


Example: Airline Security Architectures



* do cheap things engineers recommend

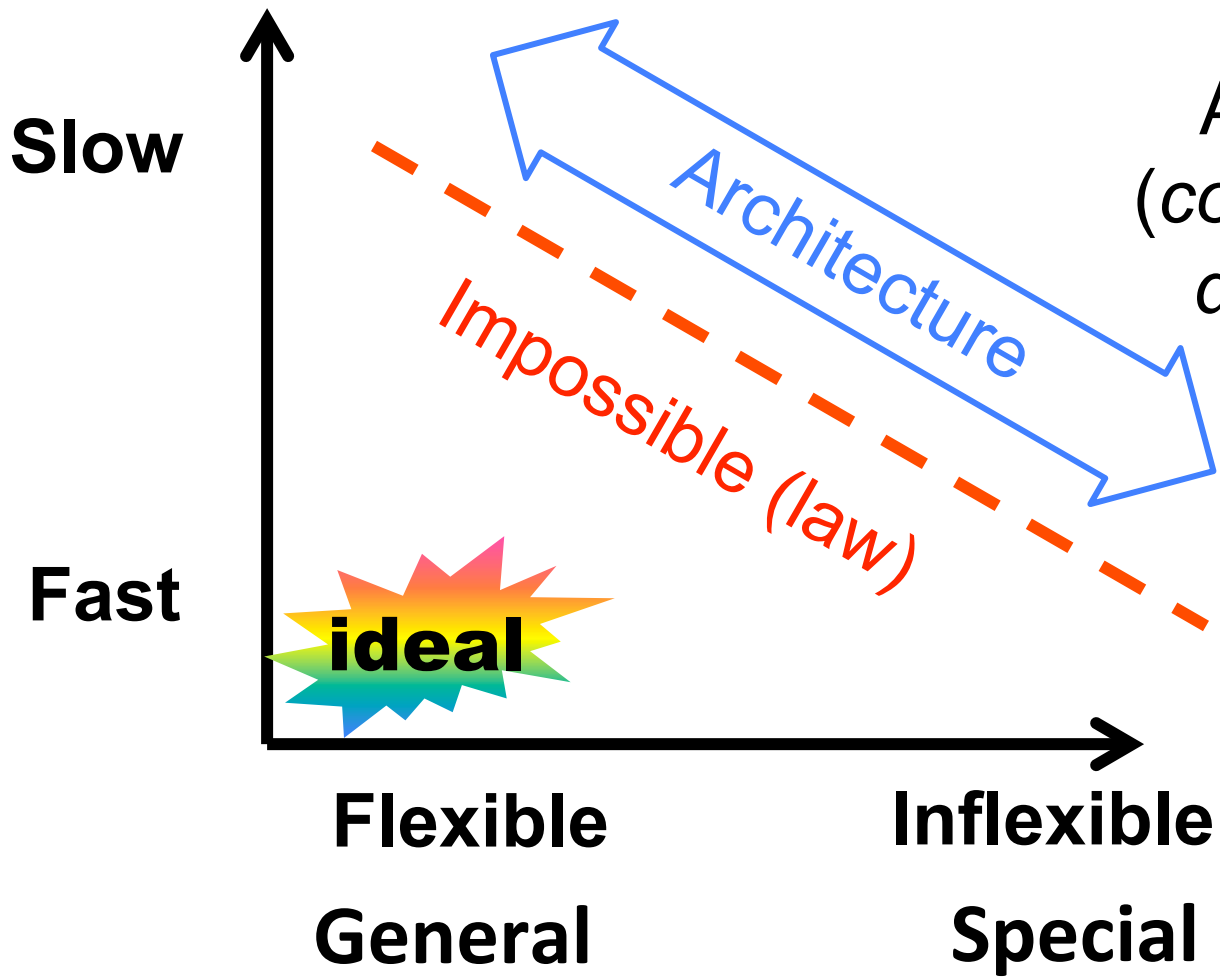
Alternatives



Original slide courtesy John Doyle

Computability Perspective

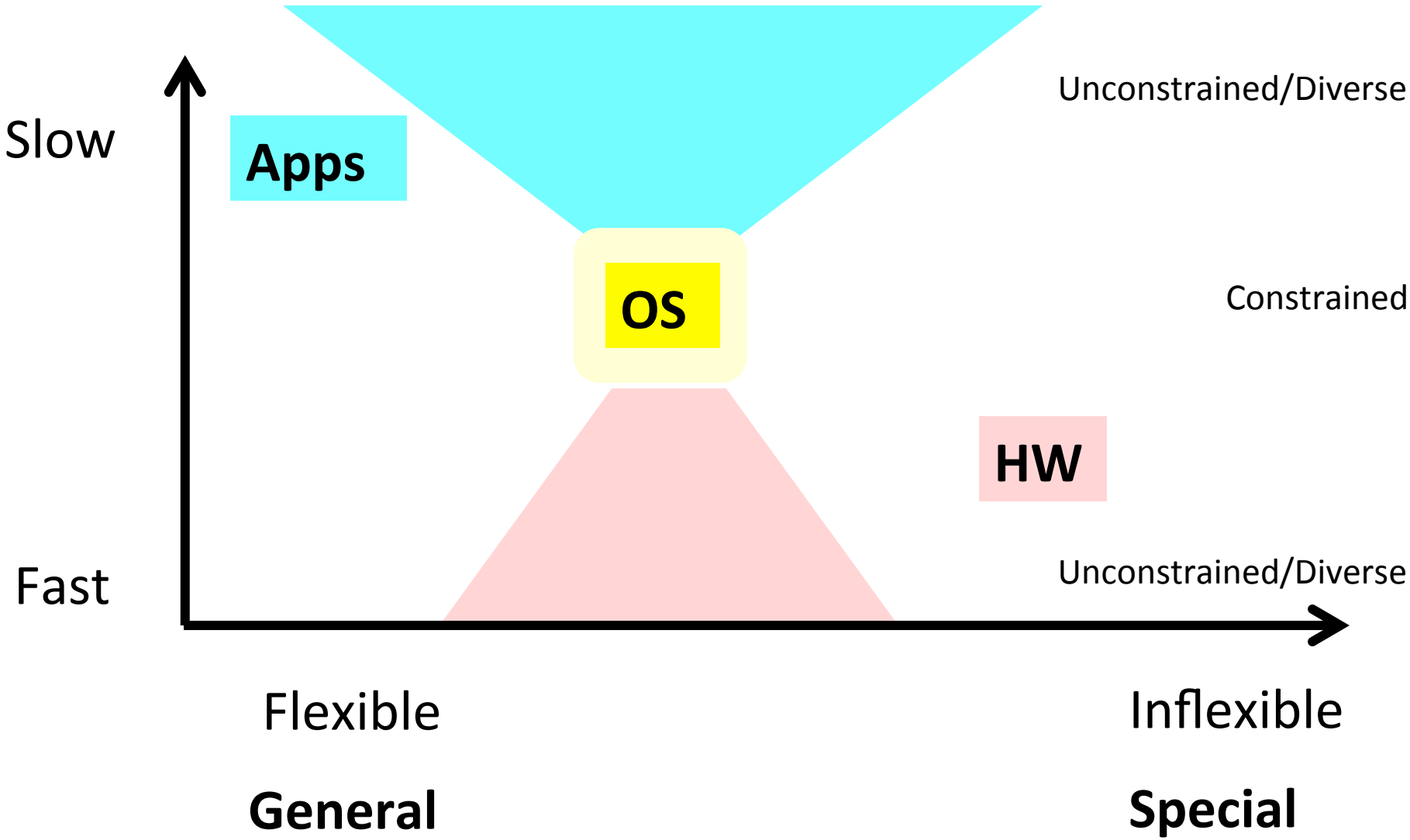
Universal Laws and Architectures (Turing)



Architecture (*constraints that deconstrain*)

Slide courtesy John Doyle

Case Study: Operating Systems Stack



RYF Behavior is found everywhere

Robust	*Yet Fragile*
☺ Metabolism	☹ Obesity, diabetes
☺ Regeneration & repair	☹ Cancer
☺ Immune/inflammation	☹ AutoImmune/Inflame
☺ Microbe symbionts	☹ Parasites, infection
☺ Neuro-endocrine	☹ Addiction, psychosis,…
🗐 Complex societies	☠ Epidemics, war,…
🗐 Advanced technologies	💣 Disasters, global &!%$#
🗐 Risk “management”	💣 Obfuscate, amplify,…

Accident or necessity?

Robust	Fragile
☺ Metabolism	☹ Obesity, diabetes
☺ Regenerati	
☺ Healing wo	une/Inflame

☹ Fat accumulation
☹ Insulin resistance
☹ Proliferation
☹ Inflammation

Same Mechanisms

- Fragility ← Hijacking, side effects, unintended…
- Of mechanisms evolved for robustness
- Complexity ← ***control***, robust/fragile tradeoffs
- Math: robust/fragile constraints ("conservation laws")

Both

Accident or necessity?

Slide courtesy John Doyle

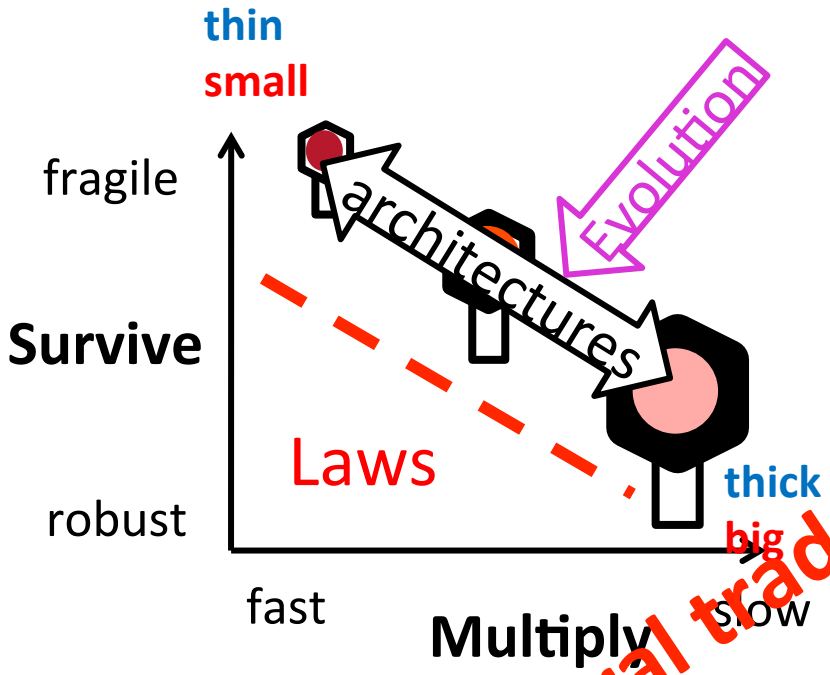
Summary: Understanding RYF is ***The*** Challenge

- It turns out that managing/understanding RYF behavior is ***the most essential challenge*** in technology, society, politics, ecosystems, medicine, etc. This means…
  - Understanding *Universal Architectural Principles*
  - Managing spiraling complexity/fragility
  - Not predicting what is likely or typical
    - But rather understanding what is catastrophic (fat tailed)
  - ***→ understanding the hidden nature of complexity***

- BTW, it is much easier to create the robust features than it is to prevent the fragilities
  - With, as mentioned, poorly understood "conservation laws"

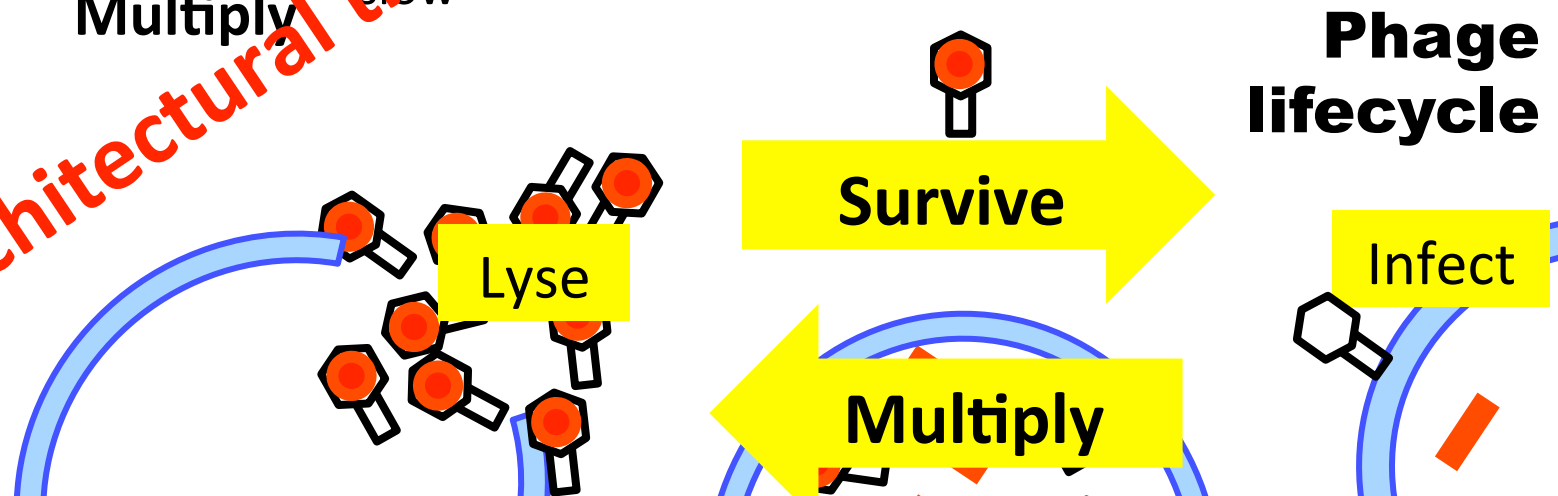
- So…
  - *What robust features (and hence fragilities) are we creating with SDN?*

So What is Universal?



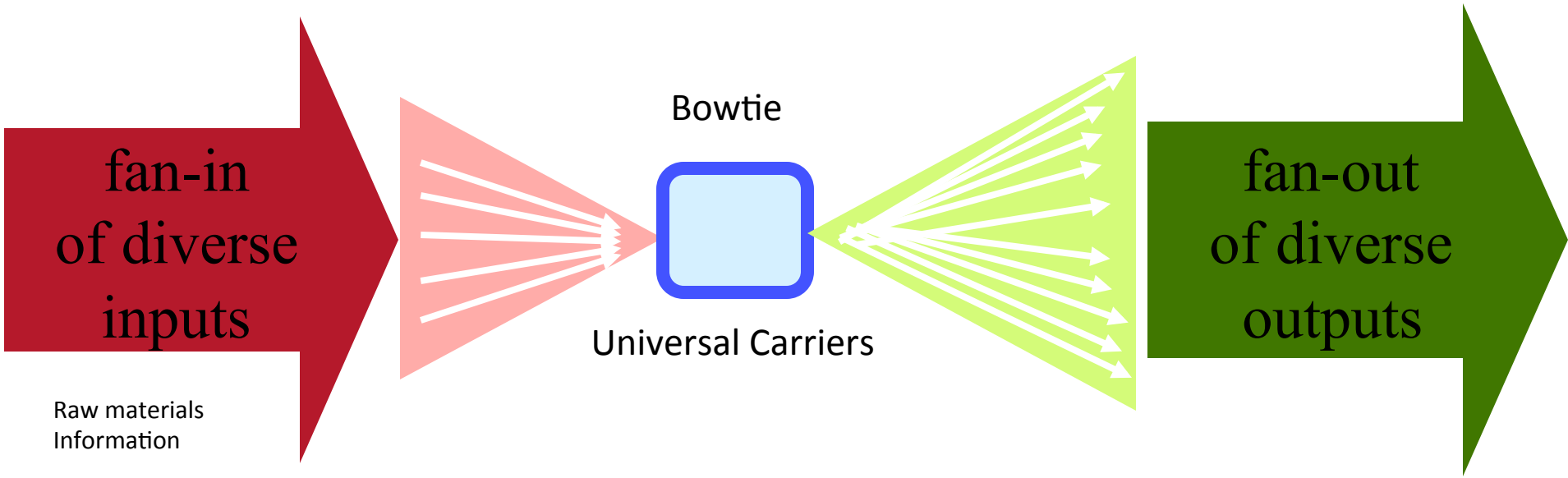
- *Laws, constraints, tradeoffs*
  - Robust/fragile
  - Efficient/wasteful
  - Fast/slow
  - Flexible/inflexible
- *Architecture*
- *Hijacking, parasitism, predation*

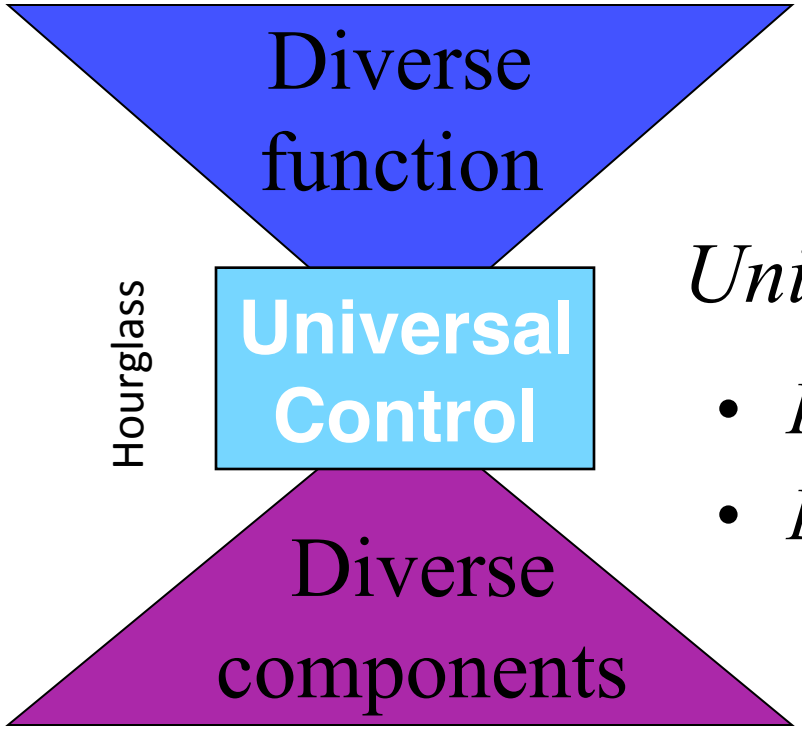
What architectural tradeoffs are we making with SDN?



Architectures

- What we have learned is that there are ***universal architectural building blocks*** found in systems that *scale* and are *evolvable*. These include
  - **RYF complexity (really a behavior)**
  - **Bowtie/Hourglass architectures**
  - **Layered Architectures**
  - Protocol Based Architectures
  - Massively distributed with *robust* control loops





Universal Architectural Principles

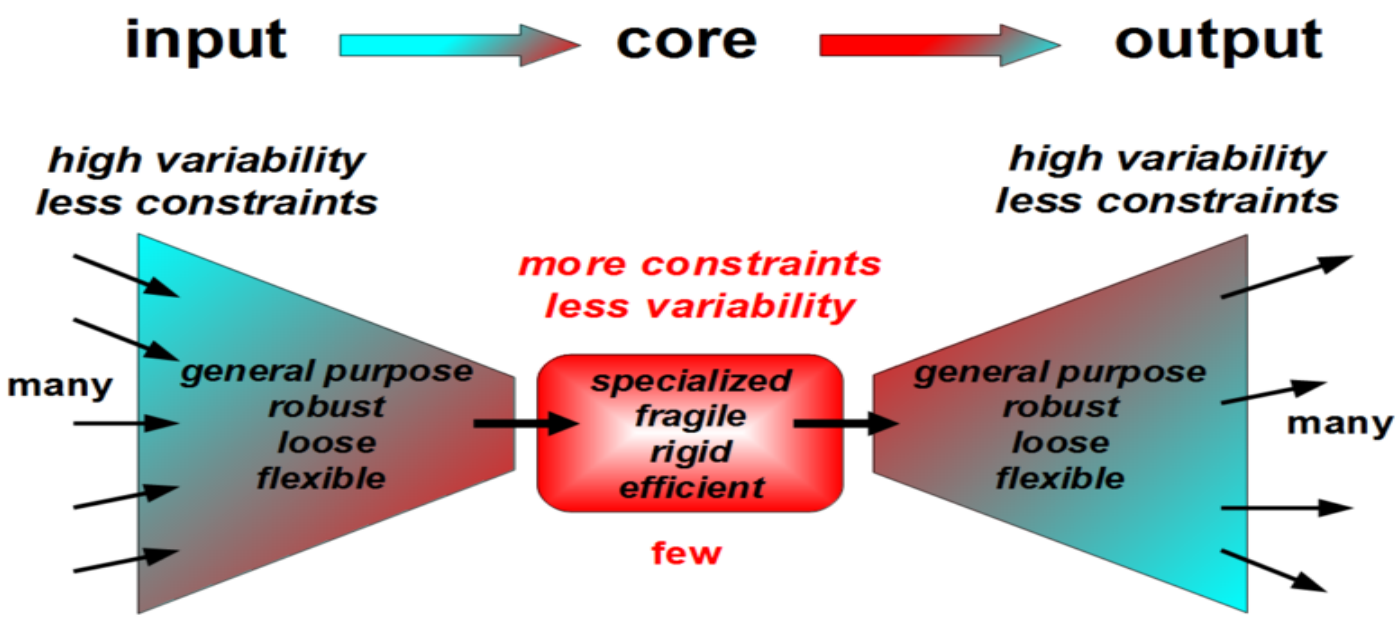
- *Hourglasses* for layering of control
- *Bowties* for flows within layers

Slide courtesy John Doyle

Bowties 101

Constraints that Deconstrain

Schematic of a "Layer"



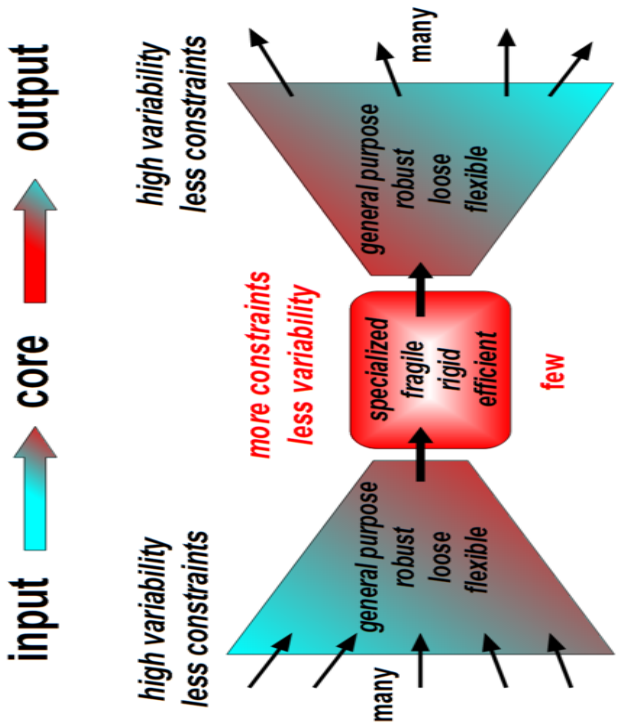
For example, the reactions and metabolites of core metabolism, e.g., *ATP metabolism*, Krebs/Citric Acid Cycle, … form a "metabolic knot". That is, ATP is a ***Universal Carrier*** for cellular energy.

1. Processes L-1 information and/or raw material flows into a "standardized" format (the L+1 abstraction)
2. Provides plug-and-play modularity for the layer above
3. Provides robustness but at the same time fragile to attacks against/using the standardized interface

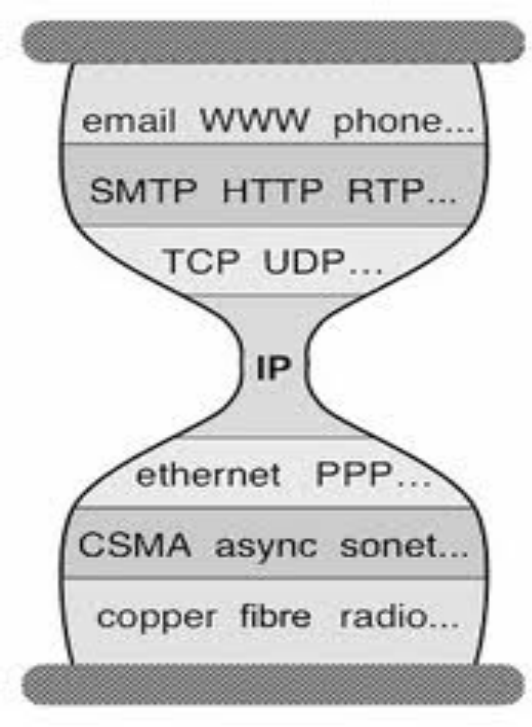
See Kirschner M., and Gerhart J., "Evolvability", Proc Natl Acad Sci USA , 95:8420–8427, 1998.

But Wait a Second

Anything Look Familiar?
(horizontal vs. vertical layering)



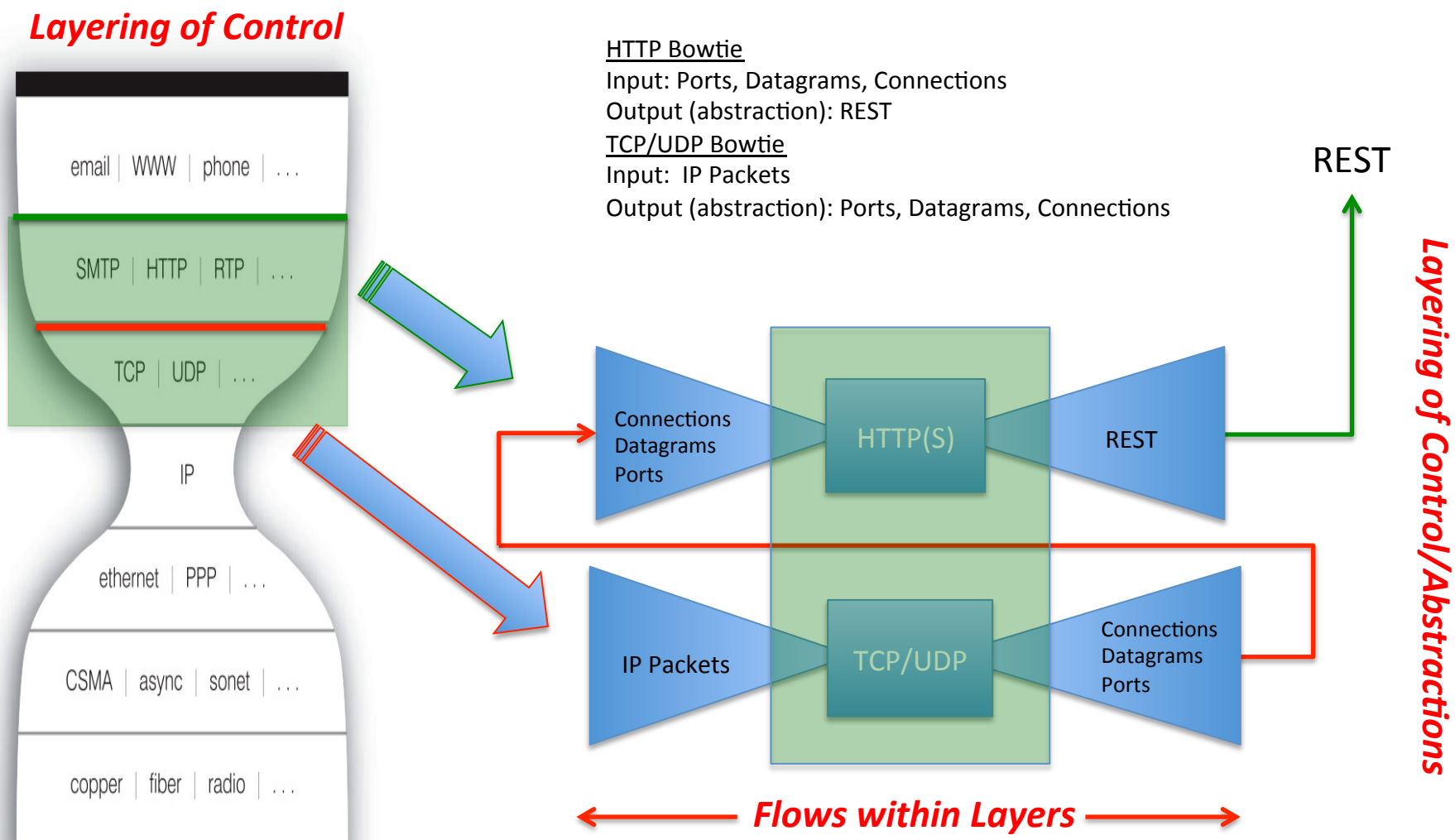
Bowtie Architecture



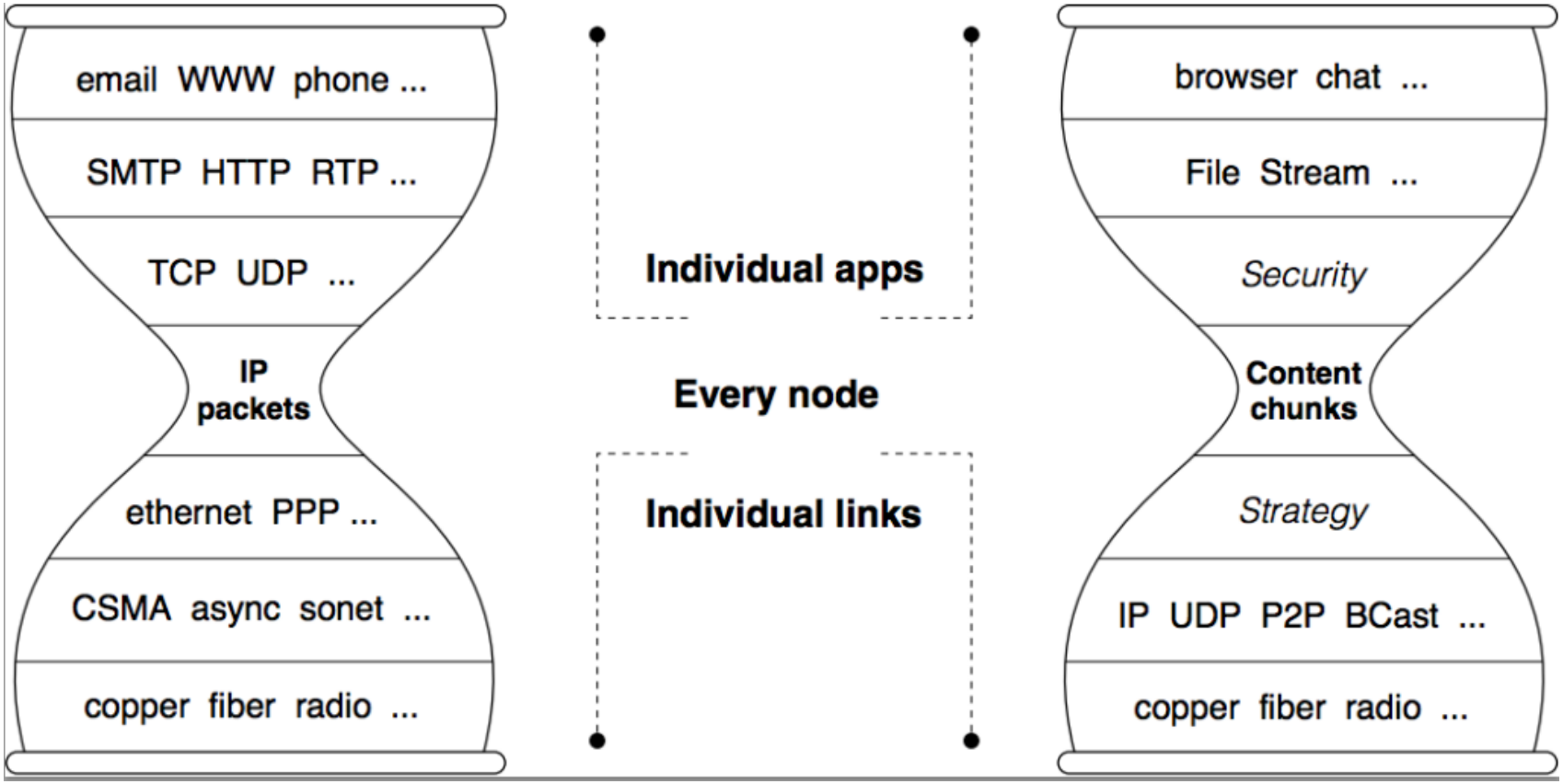
Hourglass Architecture

The *Protocol Hourglass* idea appears to have originated with Steve Deering. See Deering, S., "Watching the Waist of the Protocol Hourglass", IETF 51, 2001, http://www.iab.org/wp-content/IAB-uploads/2011/03/hourglass-london-ietf.pdf. See also Akhshabi, S. and C. Dovrolis, "The Evolution of Layered Protocol Stacks Leads to an Hourglass-Shaped Architecture", http://conferences.sigcomm.org/sigcomm/2011/papers/sigcomm/p206.pdf.

The Nested Bowtie/Hourglass Architecture of the Internet

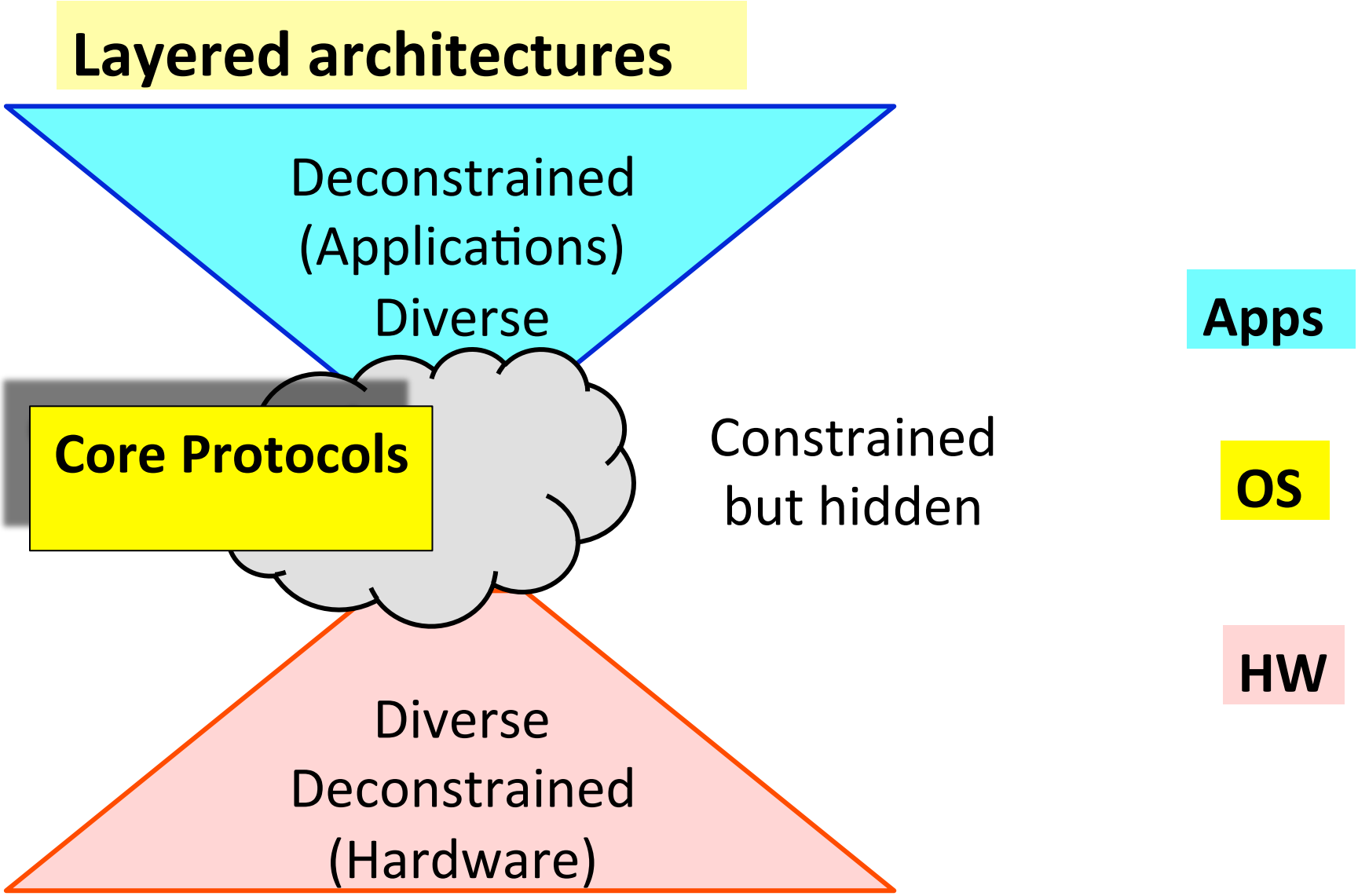


NDN Hourglass



See Named Data Networking, http://named-data.net/

Summary: Layered Architectures make *Robustness* and *Evolvability* Compatible



Mapping to Biology

Layered Bacteria

Apps

Slow
Cheap

HGT

DNA repair

Mutation

DNA replication

Transcription

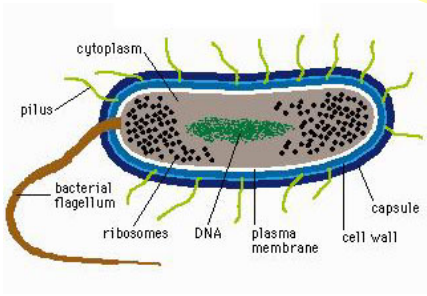
Translation

Metabolism

Signal…

OS

HW



Fast
Costly

Flexible

Inflexible

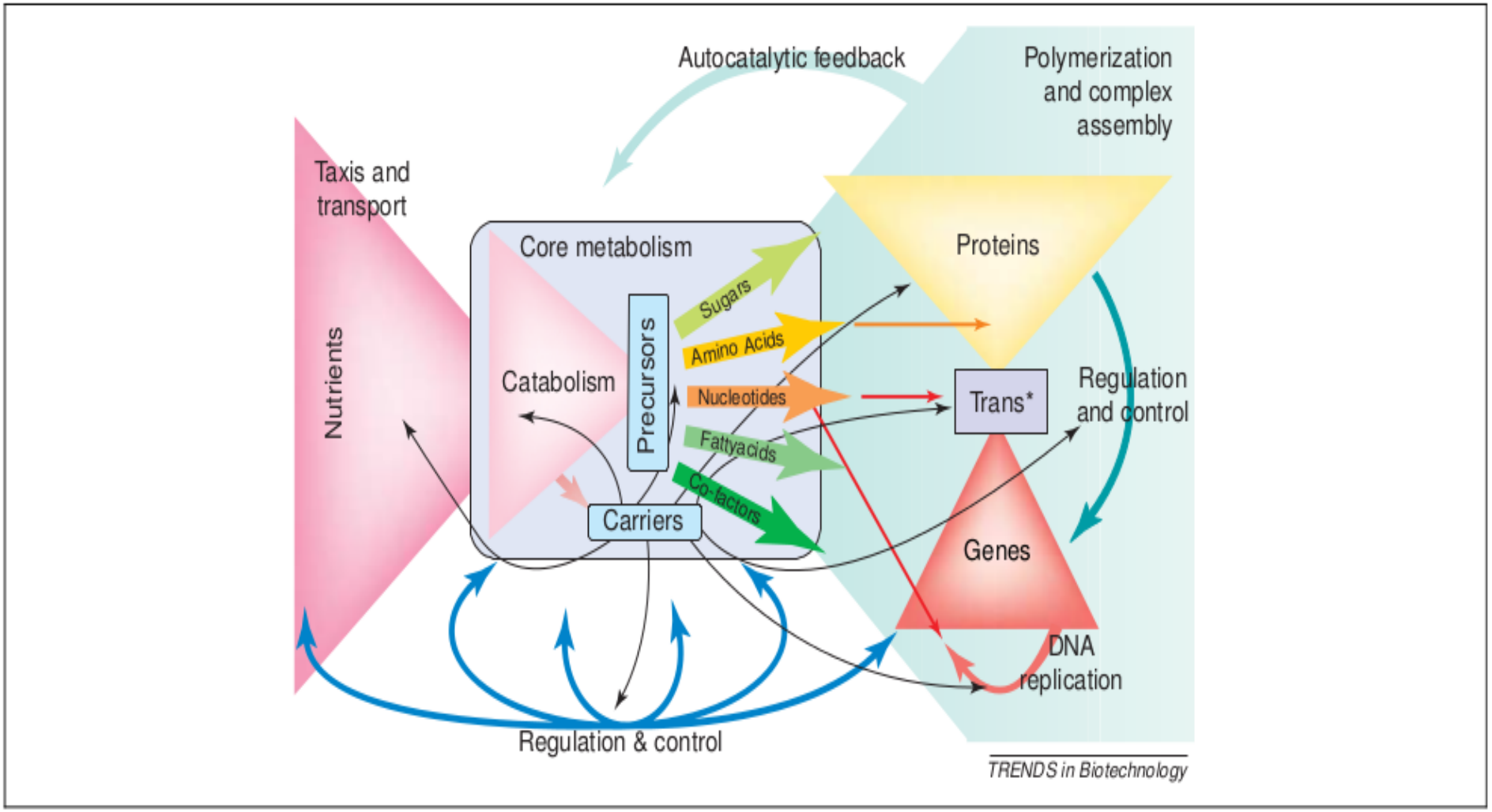
General

Special

Slide courtesy John Doyle

BTW, In Practice Things are More Complicated

The Nested Bowtie/Hourglass Architecture of Metabolism



See Csete, M. and J. Doyle, "Bowties, metabolism and disease", TRENDS in Biotechnology, Vol. 22 No. 9, Sept 2004

Agenda

- ~~Macro Trends?~~
- ~~Context: SDN Problem Space and Hypothesis~~
- ~~Complexity, SDN and Universal Principles~~
- SDN: Architecture and where we're going
- Summary and Q&A if we have time

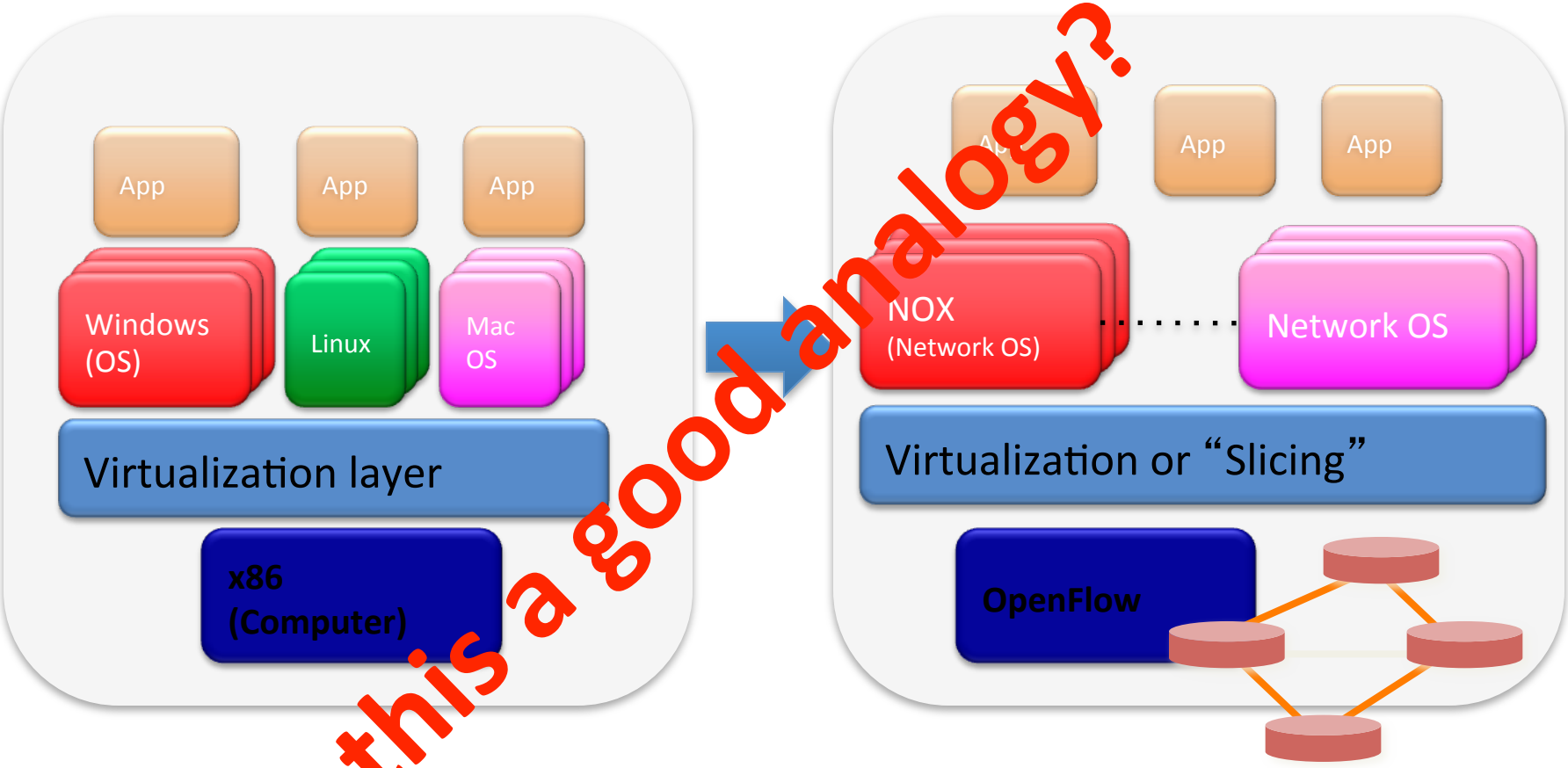
Ok, Back to SDN
How Did We Get Here?

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?

So Let's Have a Look at OF/SDN Here's Another View of the Thesis



Is this a good analogy?

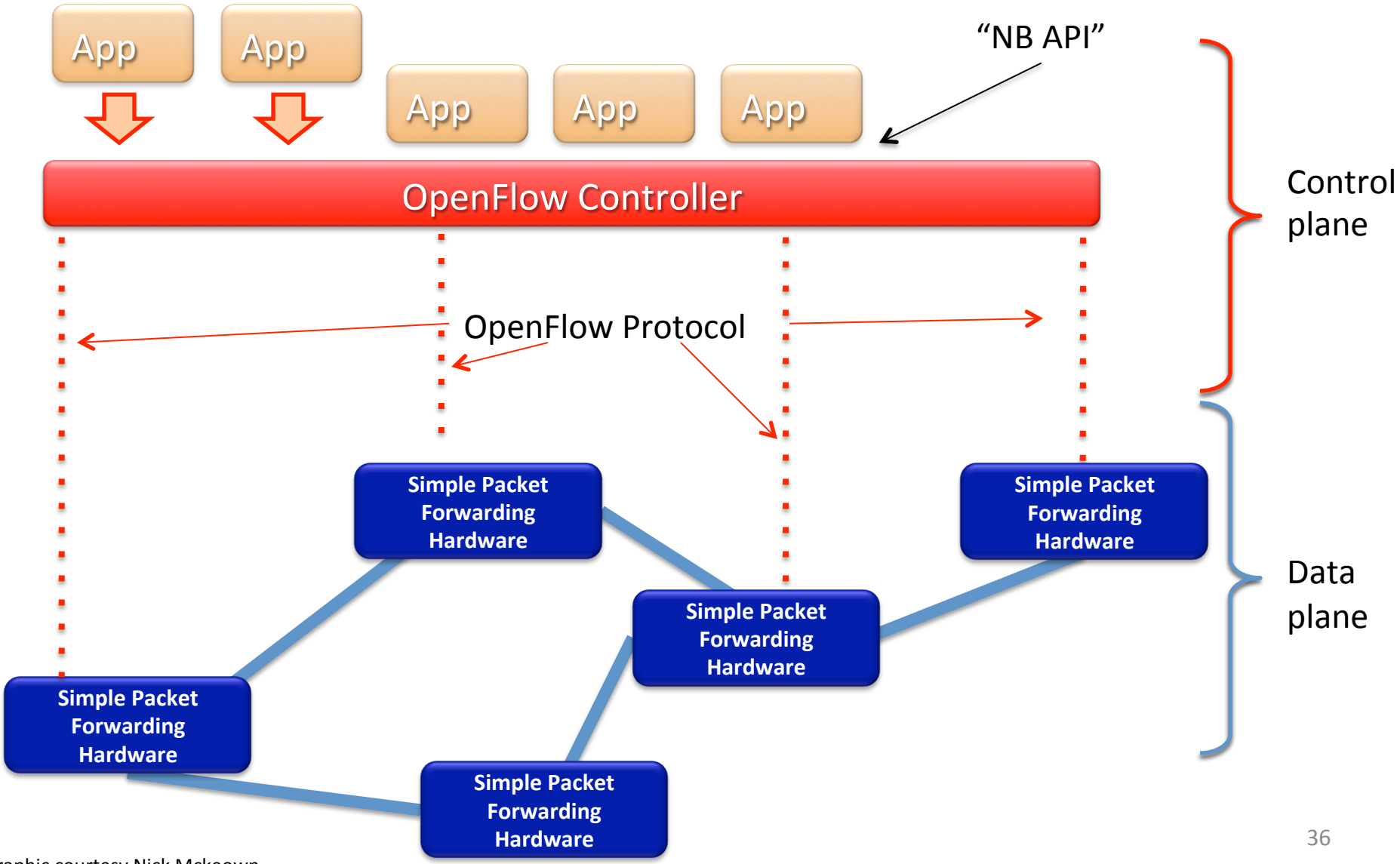
Computer Industry

Network Industry

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

Graphic Courtesy Rob Sherwood

A Closer Look



Graphic courtesy Nick Mckeown

So Does the OF/SDN-Compute Analogy Hold?

Mainframe Business Model

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

Net Equipment

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

Commodity Server

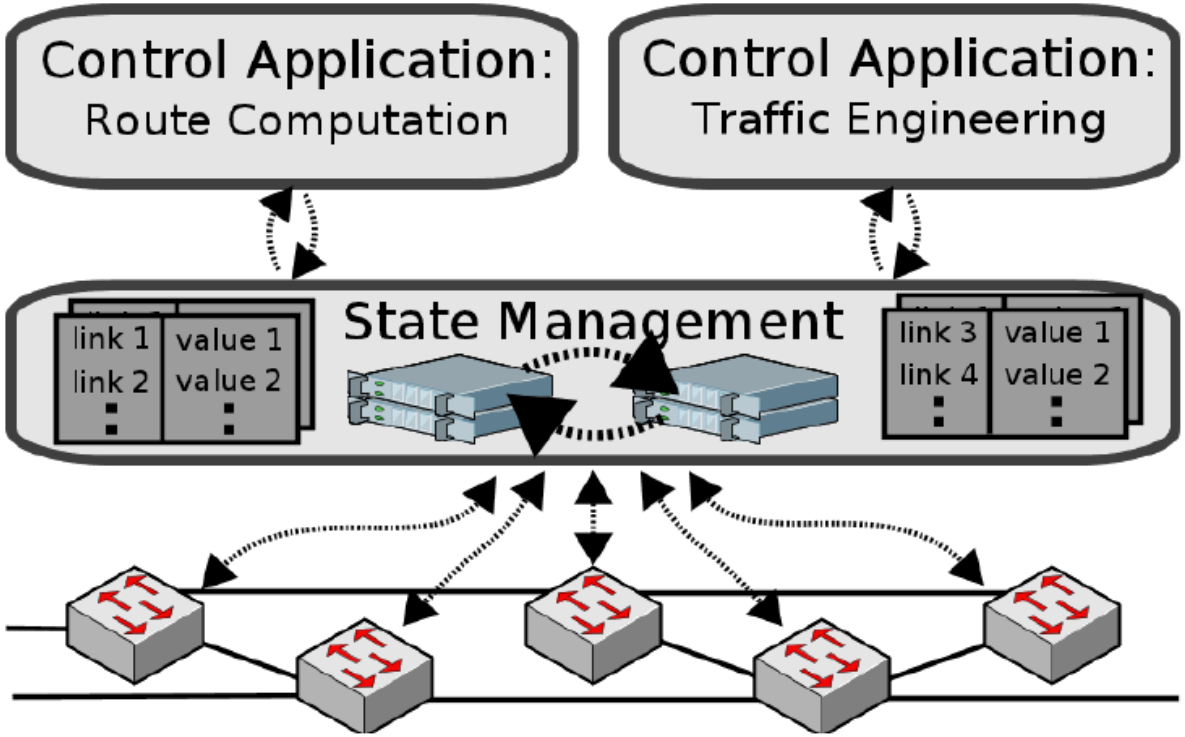
- **Example**:
  - Juniper EX 8216 (used in core or aggregation layers)
  - Fully configured list: $716k w/o optics and $908k with optics
- **Solution**: Merchant silicon, H/W independence, open source protocol/mgmt stack

Really Doesn't Look Like It

A better analogy would be an open source network stack/OS on white-box hardware

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

BTW, Logically Centralized?



Key Observation: Logically centralized → distributed system → tradeoffs between control plane convergence and state consistency model. See the *CAP Theorem.*

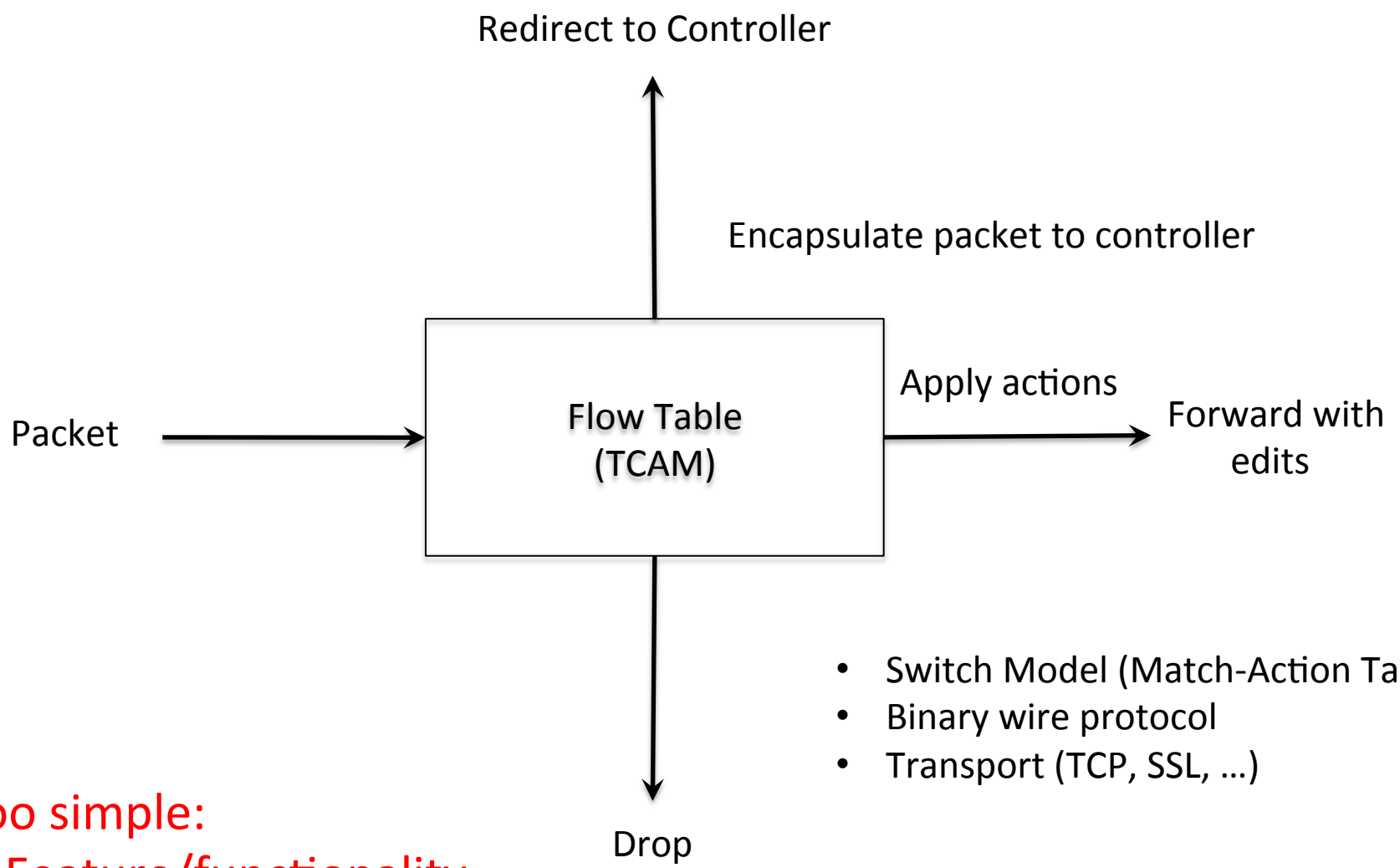
Architectural Implication: If you break CP/DP fate sharing you have to deal the following physics: $\Omega(convergence) = \Sigma\ RTT(controller, switch_i) + PPT(i,controller) + PPT(switch_i)$

Graphic courtesy Dan Levin <dlevin@net.t-labs.tu-berlin.de>

BTW, Nothing New Under The Sun…

- ***Separation of control and data planes and centralized control*** are not a new ideas. 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
  - 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)
  - …

Drilling Down: What is OpenFlow 1.0?

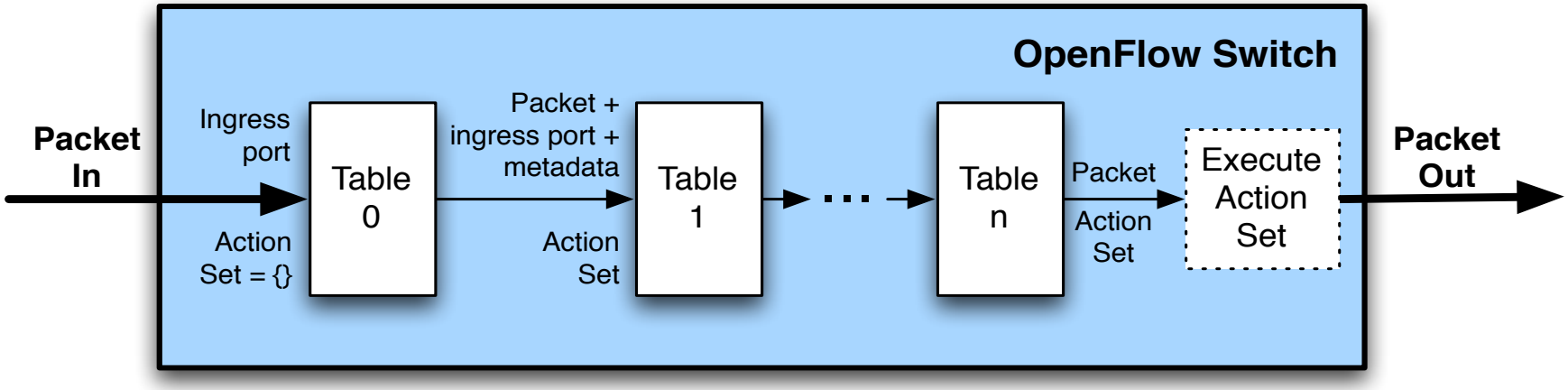


- Switch Model (Match-Action Tables)
- Binary wire protocol
- Transport (TCP, SSL, …)

Too simple:

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

OK, Fast Forward to Today: OF 1.1+



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

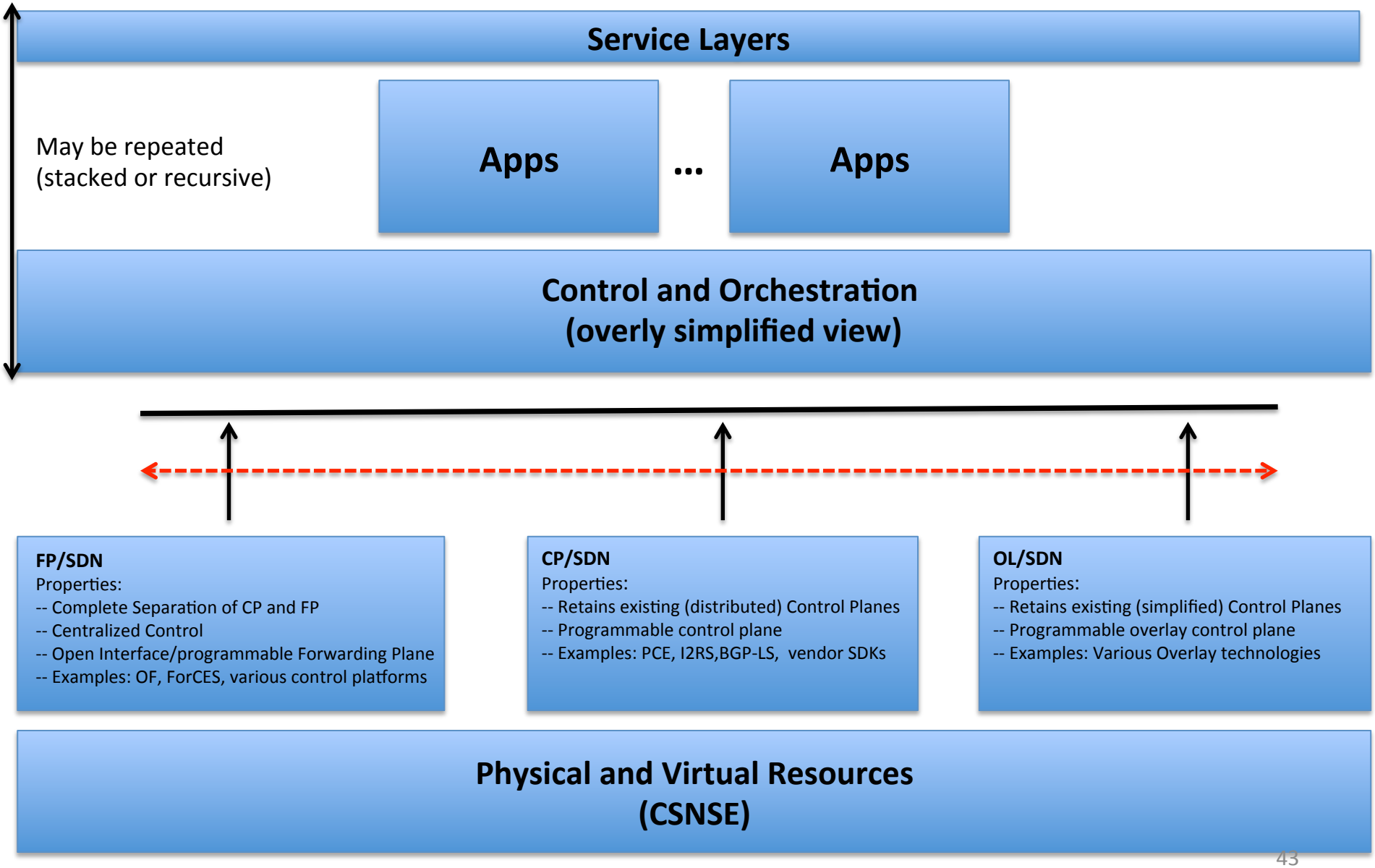
- Why this design?
  - Combinatoric explosion(s) s/a routes*policies in single table
- However, intractable complexity: *O*(n!) paths through tables of a *single switch*
  - $c \approx a^{(2^l)} + \alpha$
  - where a = number of actions in a given table, l = width of match field, and
  - α all the factors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

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

A Perhaps Controversial View

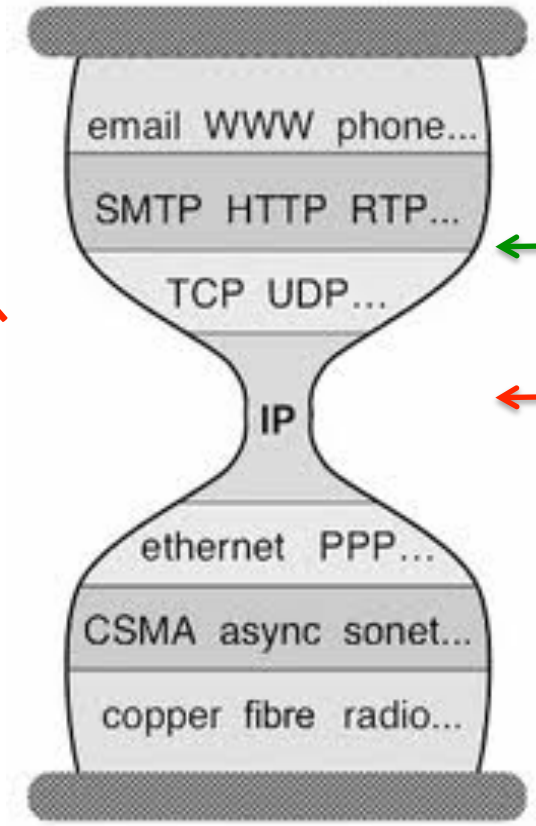
- OF/SDN is a point in a larger design space
  - But not the only one
- The larger space includes
  - Control plane programmability
  - Overlays
  - Compute, Storage, and Network Programmability
- My model: "SDN continuum"
  - http://www.ietf.org/id/draft-haleplidis-sdnrg-layer-terminology-03.txt

A Simplified View of the *SDN Continuum*



Bowties/Hourglasses?

Open Loop Control + s/w + Moore's Law →
Randomness, Uncertainty, and Volatility



OL/SDN

CP/SDN

OF/SDN?

Open Source is a wildcard

- OF/SDN?
- CP/SDN makes existing control planes programmable
- OL/SDN is an application *from the perspective of the Internet's waist*

Agenda

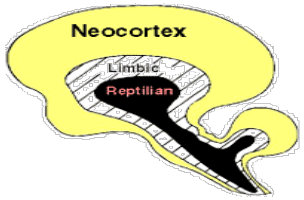
- ~~Macro Trends?~~
- ~~Context: SDN Problem Space and Hypothesis~~
- ~~Complexity, SDN and Universal Principles~~
- ~~SDN: Architecture and where we're going~~
- Summary and Q&A if we have time

So The Future: Where's it All Going?

But More Seriously….

- Current Events
  - ONF: Table Typing Patterns (TTPs) do deal with (un) tractability of OF 1.1+
  - IETF: Model Driven Everything (I2RS, …)
  - Everyone else (ETSI NFV, Cablelabs, …)
  - Open Source/*Everything*
    - http://www.opendaylight.org
    - http://www.openstack.org
    - http://opencompute.org/
  - BTW – SDN ~ DDN (DevOPs Defined Networking)
    - http://www.slideshare.net/mestery/next-gennetworkengineerskills

- Conventional Technology Curves – S & F
  - Moore's Law and the reptilian brain
    - Someone eventually has to forward packets on the wire
  - 400G and 1.2 T in the "near" term
  - Silicon photonics, denser core count, ….



- Lots of open questions
  - Architectural questions
  - Scalability and Evolvability
  - The S-word (Security)
  - "Multi-application" controllers
  - CAP theorem implications for distributed controllers
  - {AD,MD}-SAL
  - OpenFlow h/w
  - …

Q&A

Thanks!