Macro Trends, Complexity, and Software Defined Networking

David Meyer
CTO and Chief Scientist, Brocade
Director, Advanced Technology Center, University of Oregon
NANOG 58
New Orleans, Louisiana
dmm@{brocade.com,uoregon.edu,1-4-5.net,…}
http://www.1-4-5.net/~dmm/talks/nanog58.pdf

Agenda

- Macro Trends?
- Context: SDN Problem Space and Hypothesis
- SDN: How did we get here?
- Where is all of this going
  - And what role does SDN play?
- Summary and Q&A if we have time

Danger Will Robinson!!!

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

Bottom Line Here

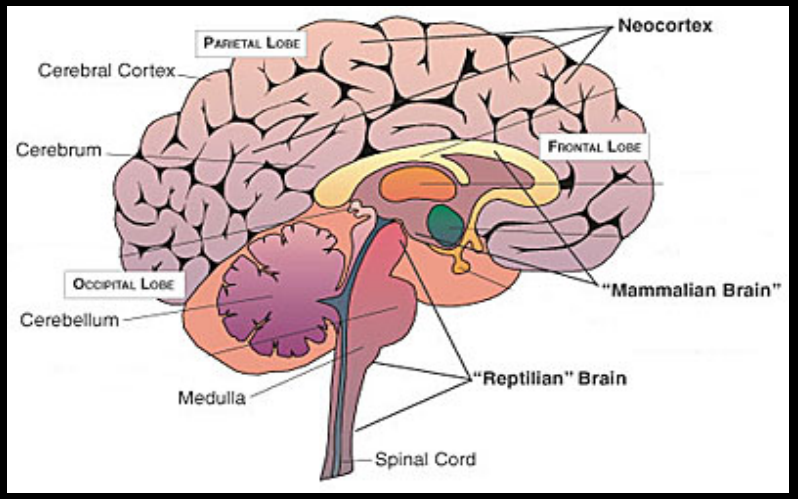
I hope to convince you that there are exist "macro trends" that are inducing uncertainty and volatility in the network space, why this is the case, how SDN (and the rise of software in general) is accelerating this effect, and finally, what we might do to take advantage of it.[1]

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

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**
  - Massively distributed control
  - Highly layered with robust control
  - Component reuse

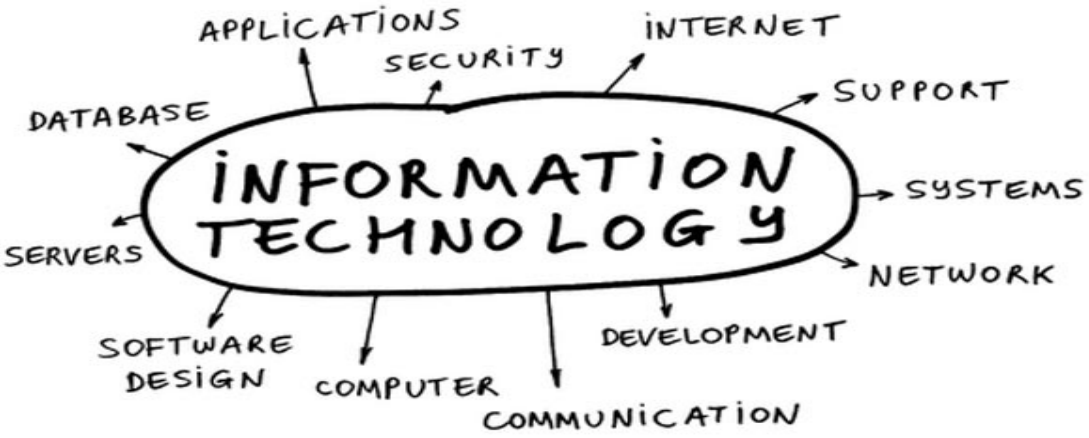
Once you have the h/w

its all about code

Trend: Everything De-silos

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

Trend: Network Centric to IT Centric



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

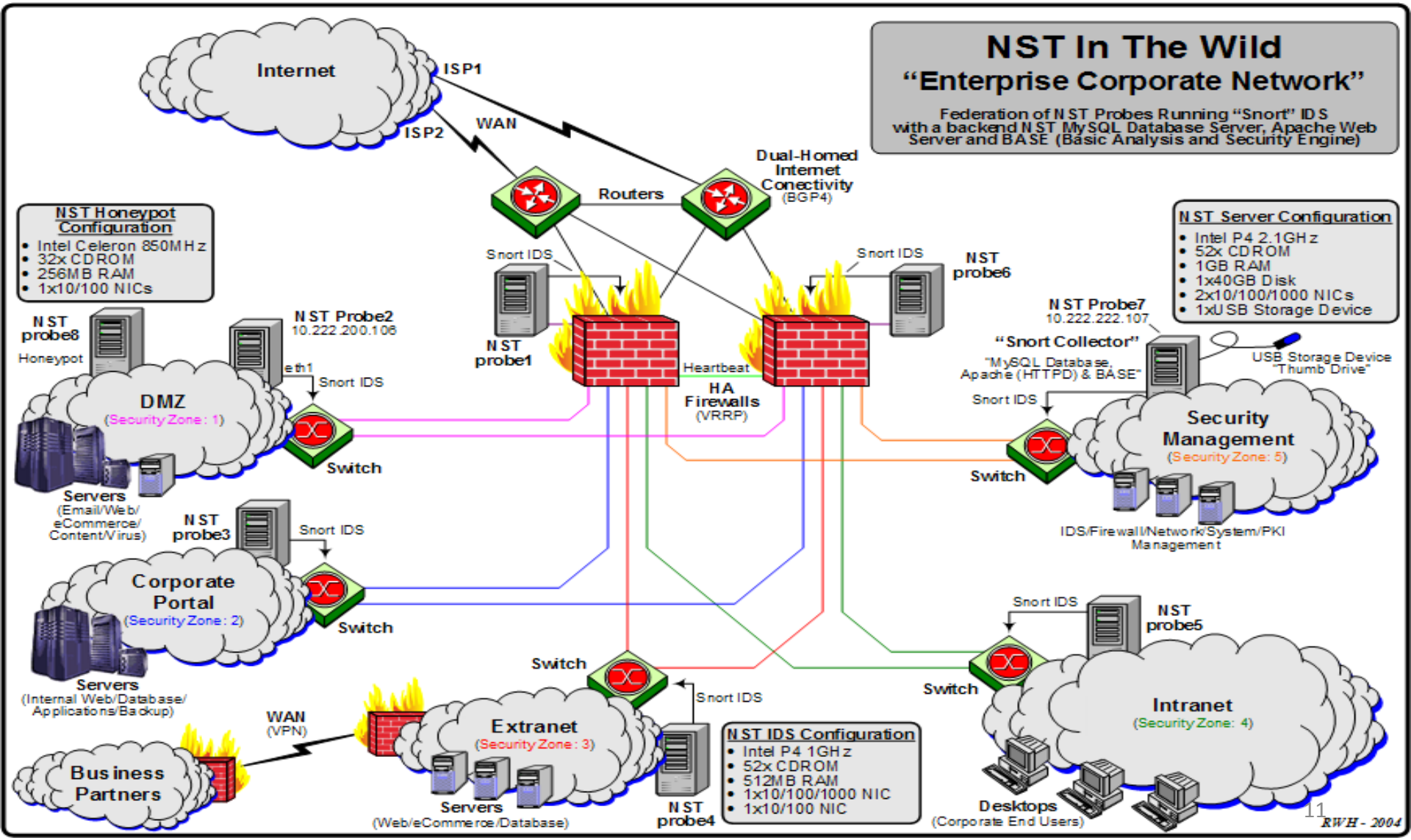
Other Important Macro Trends

- 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

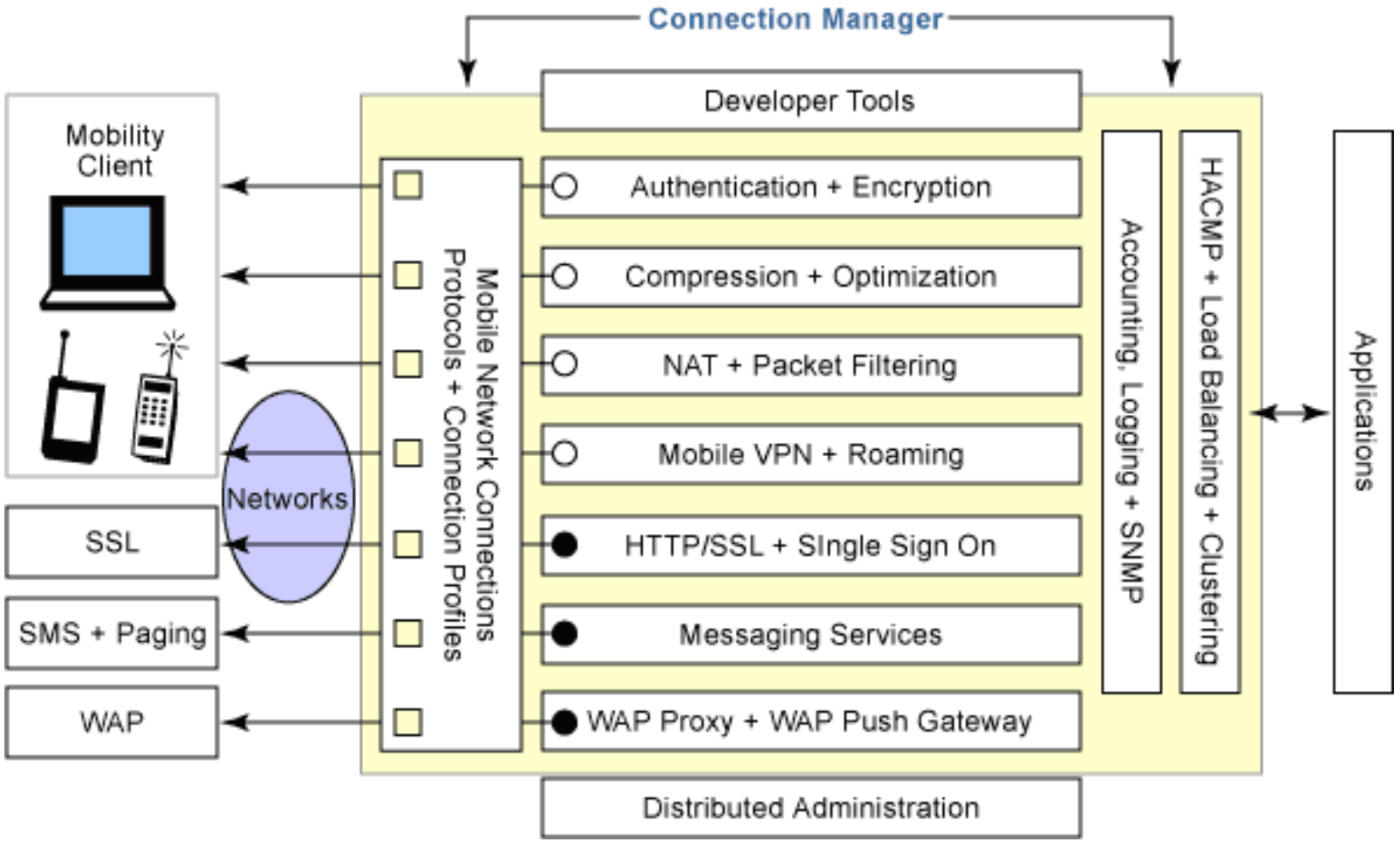
Oh Yeah, This Talk Was Supposed To Have Something To Do With SDN

- Well then, 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?



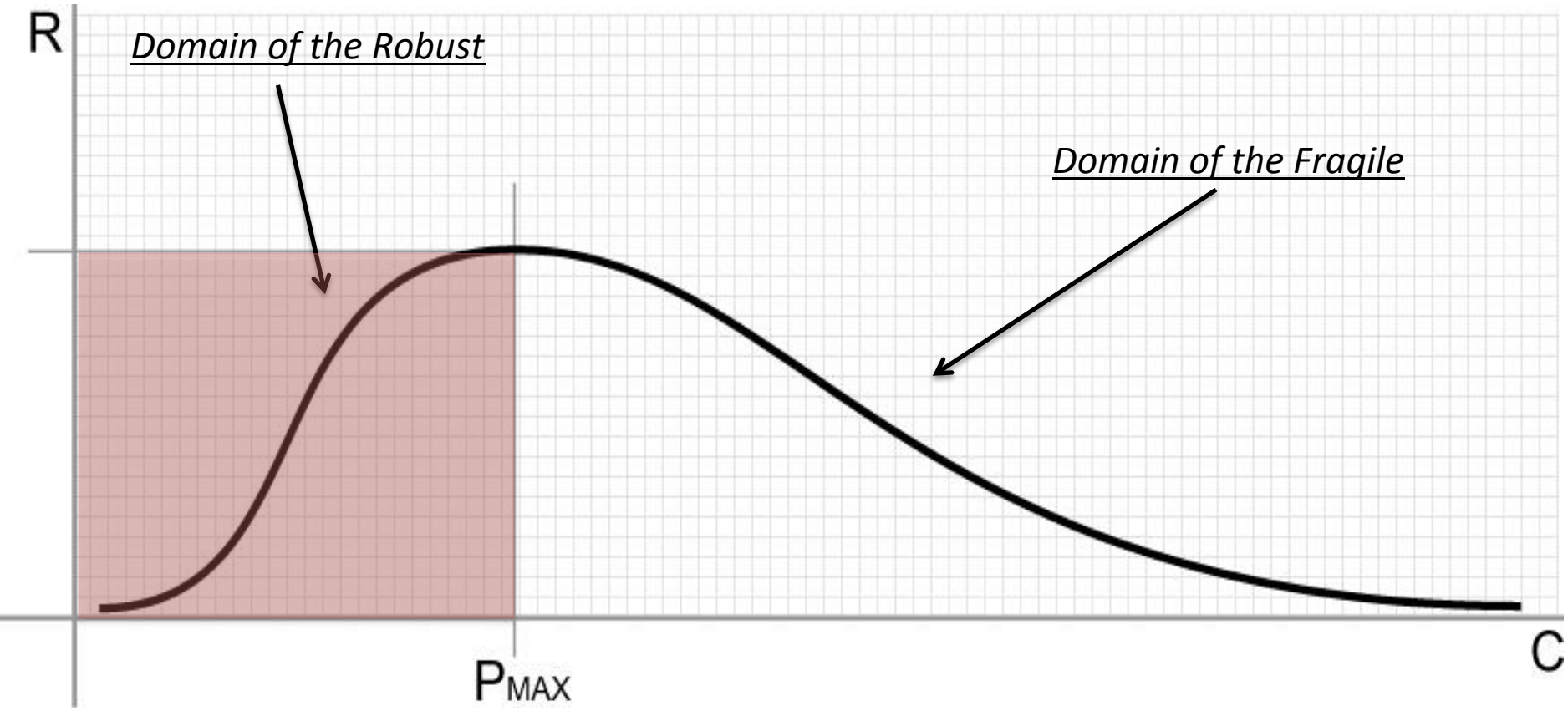
Or This?



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

Network is Robust *and* Fragile

Robustness vs. Complexity
Systems View



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

Can we characterize the Robust and the Fragile?

So what are 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) and the curve is concave
  - A little more on this later…

- 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)**
  - 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
  - Example: VRRP provides robustness to failure of a router/interface, but introduces fragilities in the protocol/ implementation
  - Complexity/Robustness Spirals

- Conjecture: The *RYF tradeoff* is a hard limit

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

RYF Examples

Robust	Yet Fragile
☺ Efficient, flexible metabolism	☹ Obesity and diabetes
☺ Complex development	☹ Rich microbe ecosystem
☺ Immune systems	☹ Inflammation, Auto-Im.
☺ Regeneration & renewal	☹ Cancer
Complex societies	Epidemics, war, …
Advanced technologies	Catastrophic failures

- "Evolved" mechanisms for robustness *allow for,* even *facilitate,* novel, severe fragilities elsewhere
- Often involving hijacking/exploiting the same mechanism
  - We've certainly seen this in the Internet space
  - Consider DDOS of various varieties
- There are hard constraints (i.e., theorems with proofs)

System features cast as Robustness

- **Scalability** is robustness to changes to the size and complexity of a system as a whole
- **Evolvability** is robustness of lineages to changes on 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
- In our case: holds for protocols, systems, and operations

Brief Aside: 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***

- 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 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
    - Consider: 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 (e.g., outside say, 10 σ)
  - Again, ARP storms, congestion collapse, AS 7007, DDOS, … → non-linear damage

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
  - **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)
  - But see Jim's blog
  - http://www.renesys.com/blog/2013/05/syrian-internet-fragility.shtml

Aside: What 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.***"

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

BTW, This Might Also Obvious But…

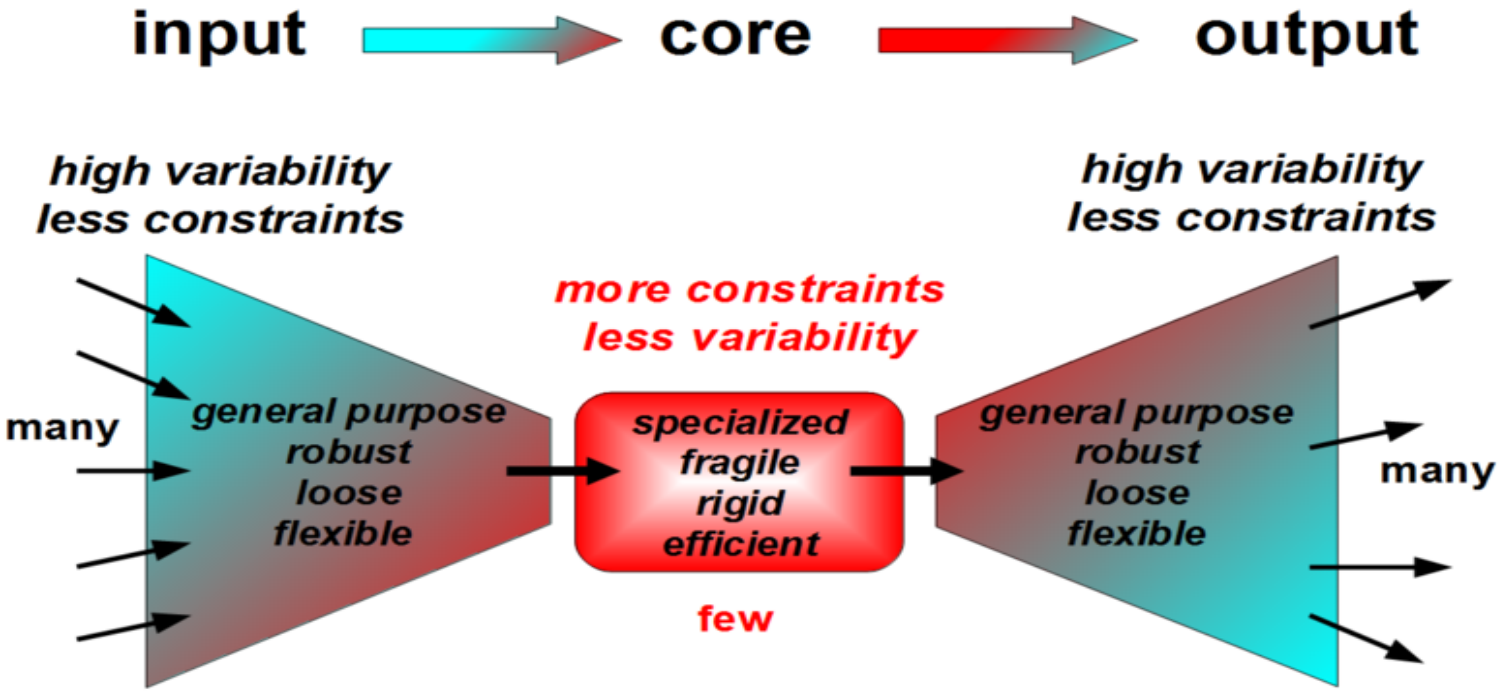
- Networks are incredibly general and expressive structures
  - *G = (V,E)*
- Networks are extremely common in nature
  - Immune systems, energy metabolism, transportation systems, health care systems, *Internet*, macro economies, forest ecology, the main sequence (stellar evolution), galactic structures, ….
  - "Almost everything you see can be explained as either a network and/or a queue"
- So it comes as no surprise that we study, for example, biological systems in our attempts to get a deeper understanding of complexity and the architectures that provide for scalability, evolvability, and the like
- Ok, this is cool, but what are the key architectural takeaways from this work for us ?
  - where *us \in {ops, engineering, architects …}*
  - And how might this effect the way we build and operate networks?
  - Keep this question in mind…

Ok, Key Architectural Takeaways?

- What we have learned is that there are ***fundamental architectural building blocks*** found in systems that scale and are evolvable. These include
  - **RYF complexity**
  - **Bowtie architectures**
  - Massively distributed with *robust* control loops
    - Contrast optimal control loops and hop-by-hop control
  - Highly layered
    - But with layer violations, e.g., Internet, overlay virtualization
  - Protocol Based Architectures (PBAs)
  - Degeneracy

Bowties 101

Constraints that Deconstrain

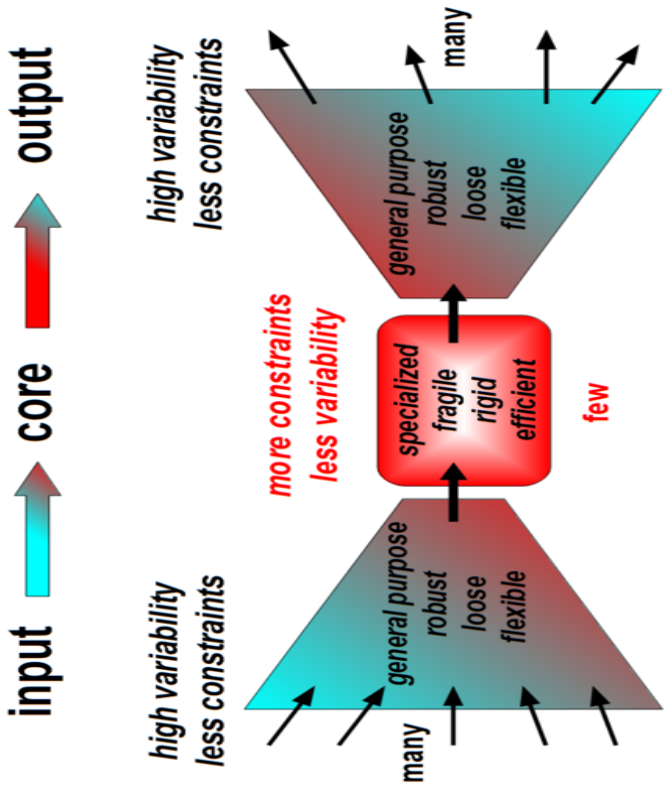


For example, the reactions and metabolites of core metabolism, e.g., *ATP metabolism*, Krebs/Citric Acid cycle signaling networks, …

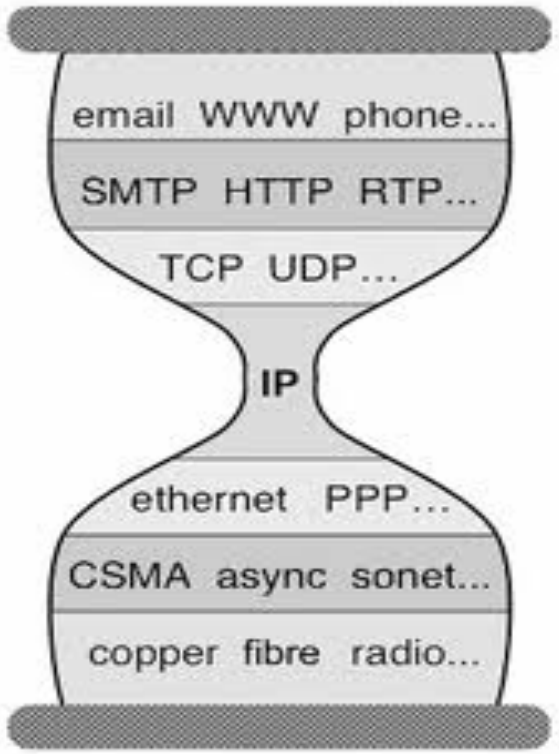
See Kirschner M., and Gerhart J., "Evolvability", Proc Natl Acad Sci USA , 95:8420–8427, 1998.

But Wait a Second

Anything Look Familiar?



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.

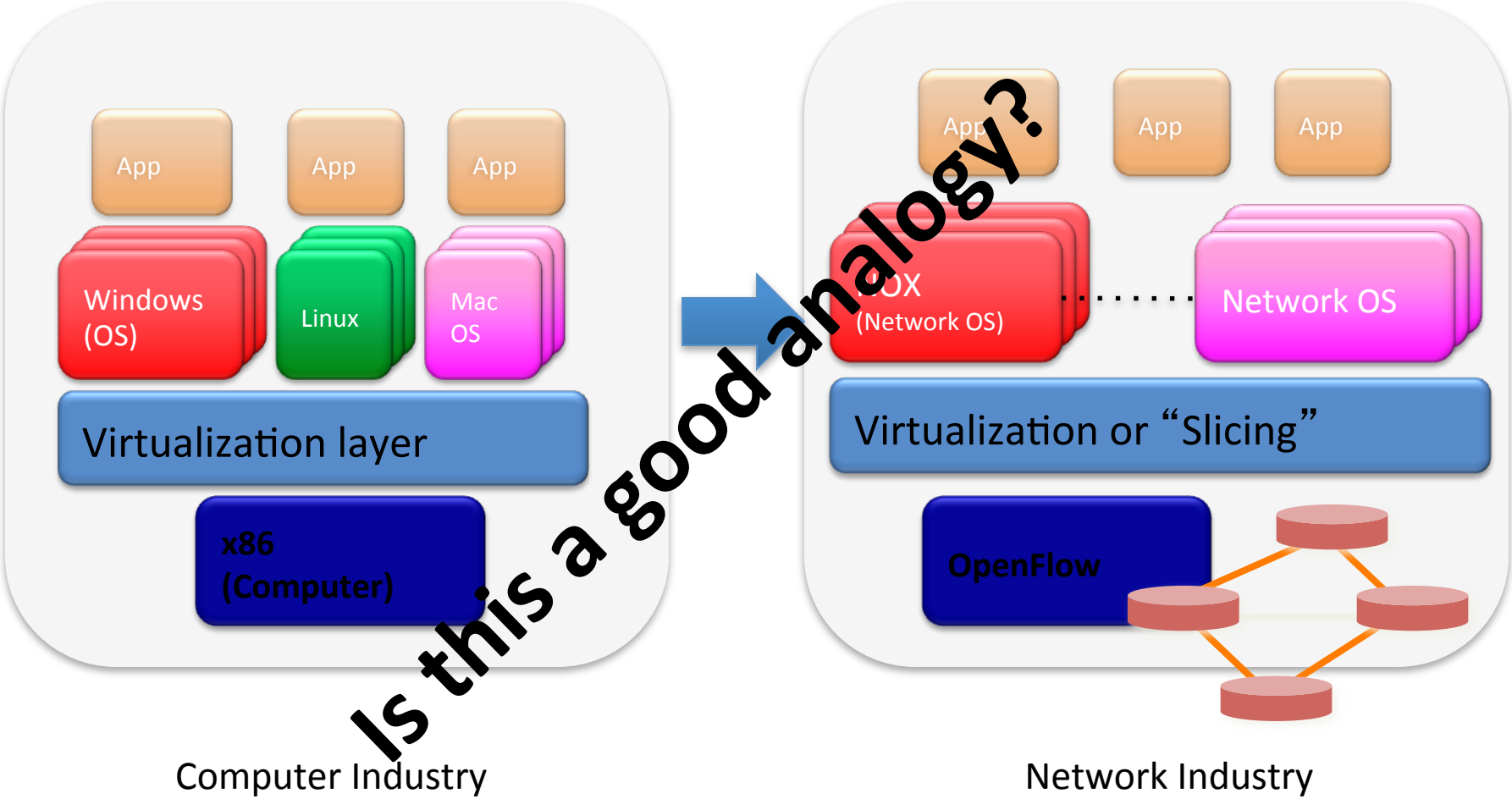
Ok, Back to SDN
How Did We Get Here?

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?

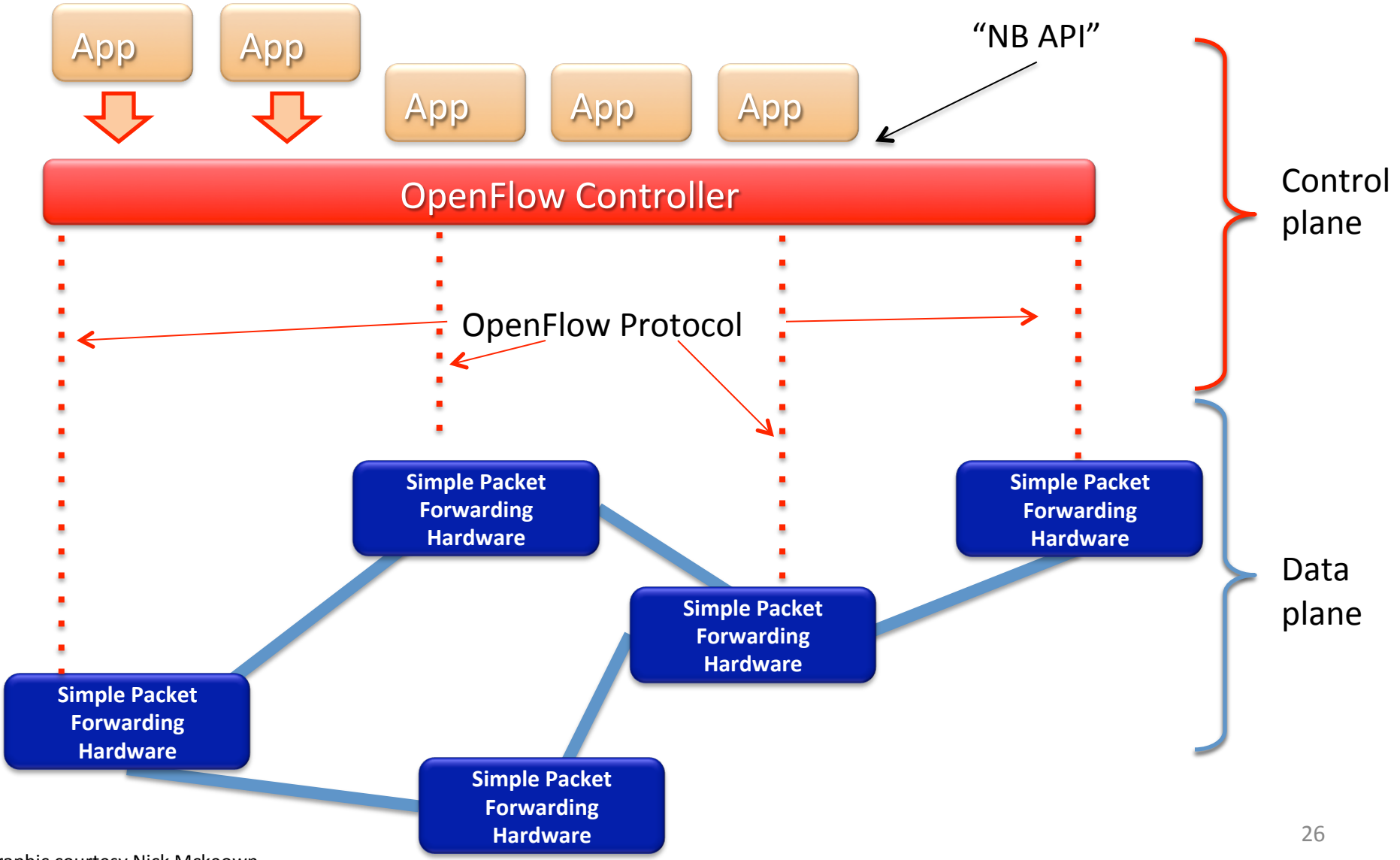
So Let's Have a Look at OF/SDN Here's Another View of the Thesis



- 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

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

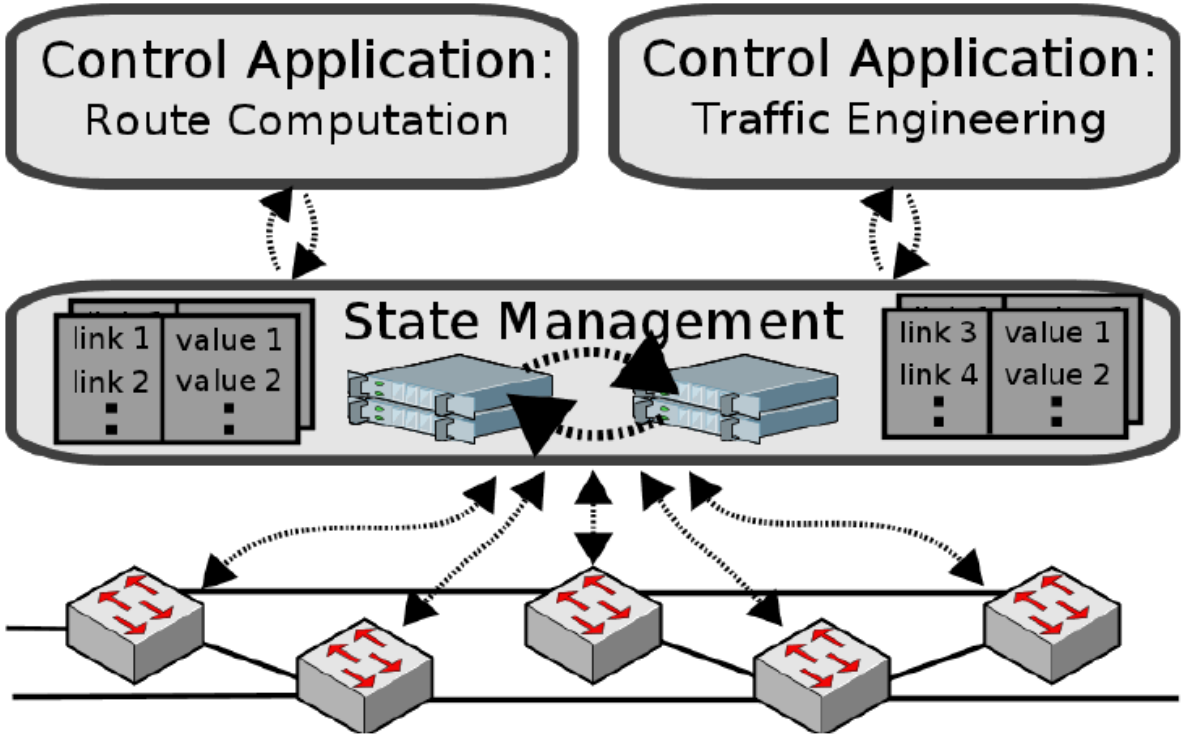
- **Example:**
  - Juniper EX 8216 (used in core or aggregation layers)
  - Fully configured list: $716k w/o optics and $908k with optics
- **Solution**: Merchant silicon, H/W independence, open source protocol/mgmt stack

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.*

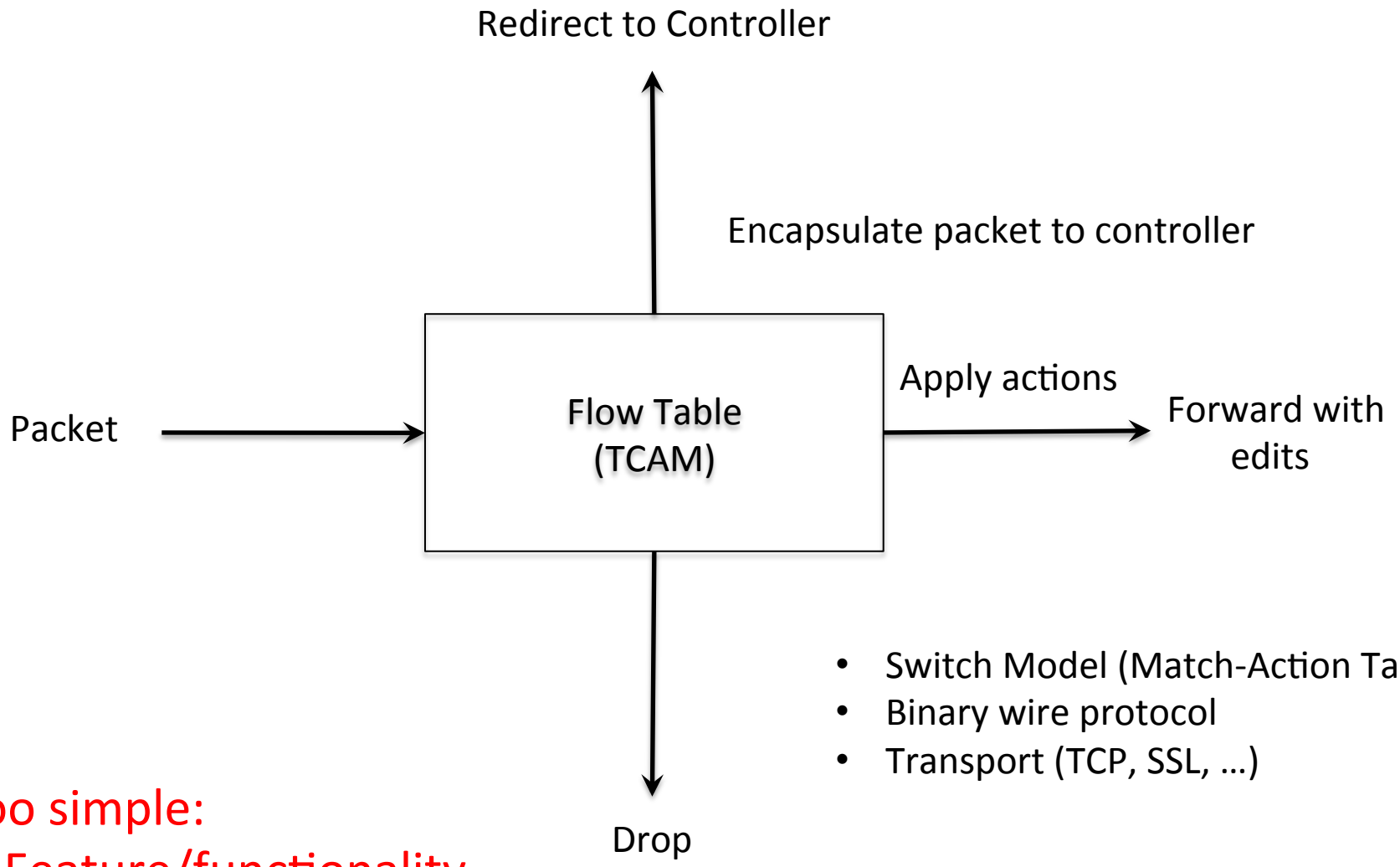
Architectural Implication: If you break CP/DP fate sharing you have to deal the following physics: $\Omega(convergence) = \Sigma\ RTT(controller, switch_i) + PPT(i,controller) + PPT(switch_i)$

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?

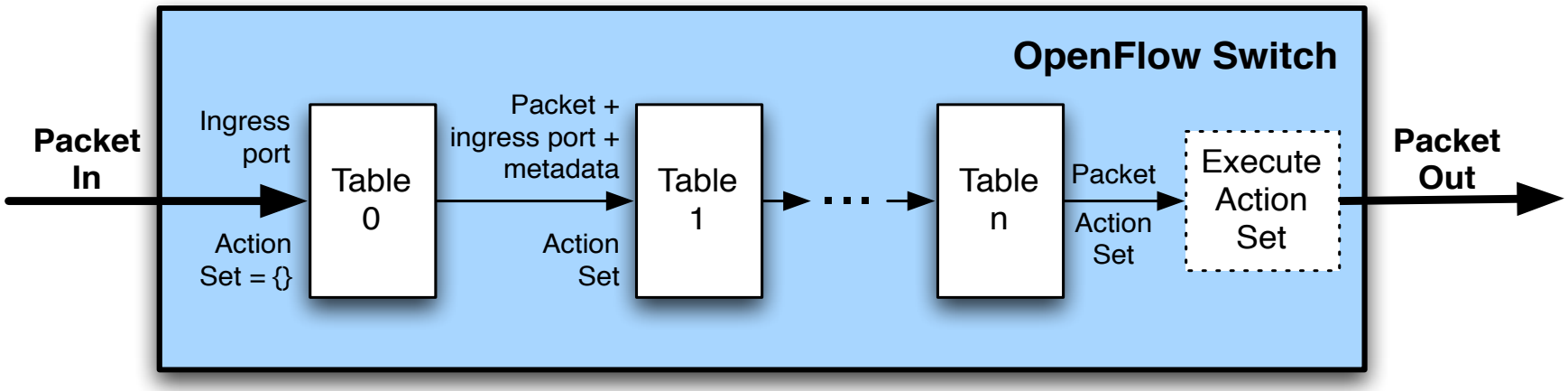


- Switch Model (Match-Action Tables)
- Binary wire protocol
- Transport (TCP, SSL, …)

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

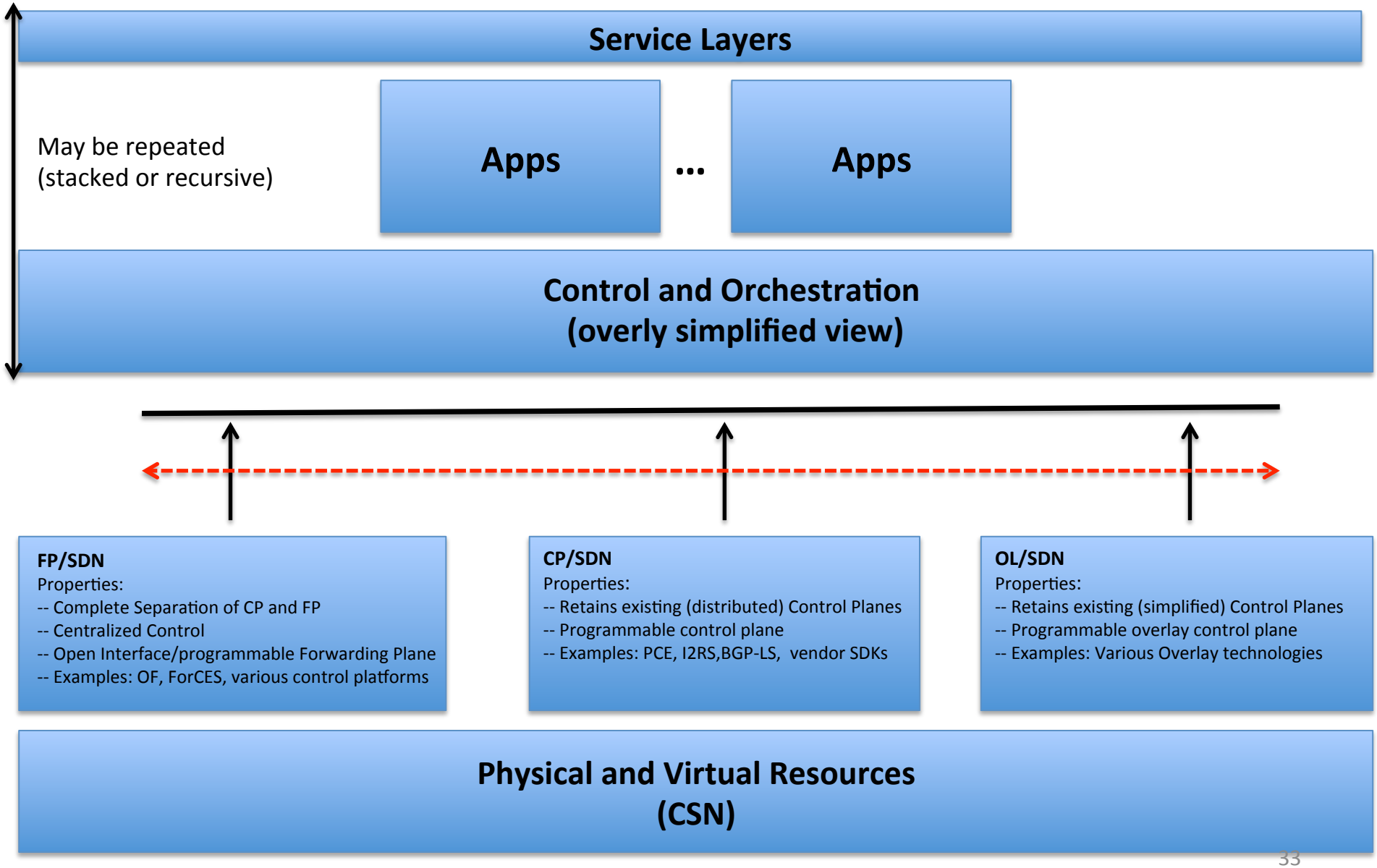
- Why this design?
  - Combinatoric explosion(s) s/a routes*policies in single table
- However, intractable complexity: *O(*n!) paths through tables of a *single switch*
  - $c \approx a^{(2^{\wedge}l)} + \alpha$
  - where a = number of actions in a given table, l = width of match field, and
  - α all the factors I didn't consider (e.g., table size, function, group tables, 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

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

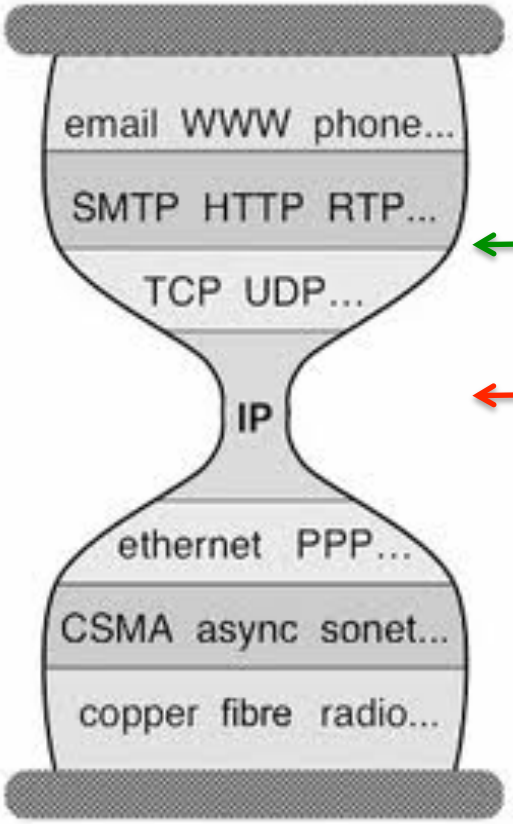
A Perhaps Controversial View

- 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"

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?

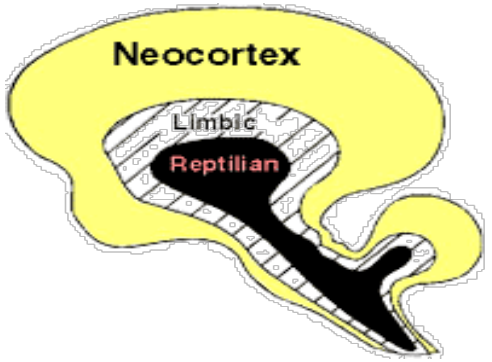
- OF/SDN?
- CP/SDN makes existing control planes programmable
- OL/SDN is an application *from the perspective of the Internet's waist*

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 photonics, 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***

- Open *everything*

Summary – What are our Options

- 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).

Q&A

Thanks!