On The Hidden Nature of Complex Systems

Tradeoffs, Architecture, Nets, Grids, Bugs, Brains, and the Meaning of Life

David Meyer
CTO and Chief Scientist, Brocade
Director, Advanced Technology Center, University of Oregon
NANOG 61
June 02-04 2014
Bellevue, WA
dmm@{brocade.com,uoregon.edu,1-4-5.net,…}
http://www.1-4-5.net/~dmm/talks/nanog61.pptx

Agenda

- Too many words, too many slides ☺
  - This talk is about thinking about networking in new ways
- Motivation and Goals for this Talk
- What is Complexity and Why is it Hidden
  - Robustness, Fragility, and Complexity
- The Architecture of Complex Systems
  - Universal Architectural Principles
- A Few Conclusions and Q&A if we have time

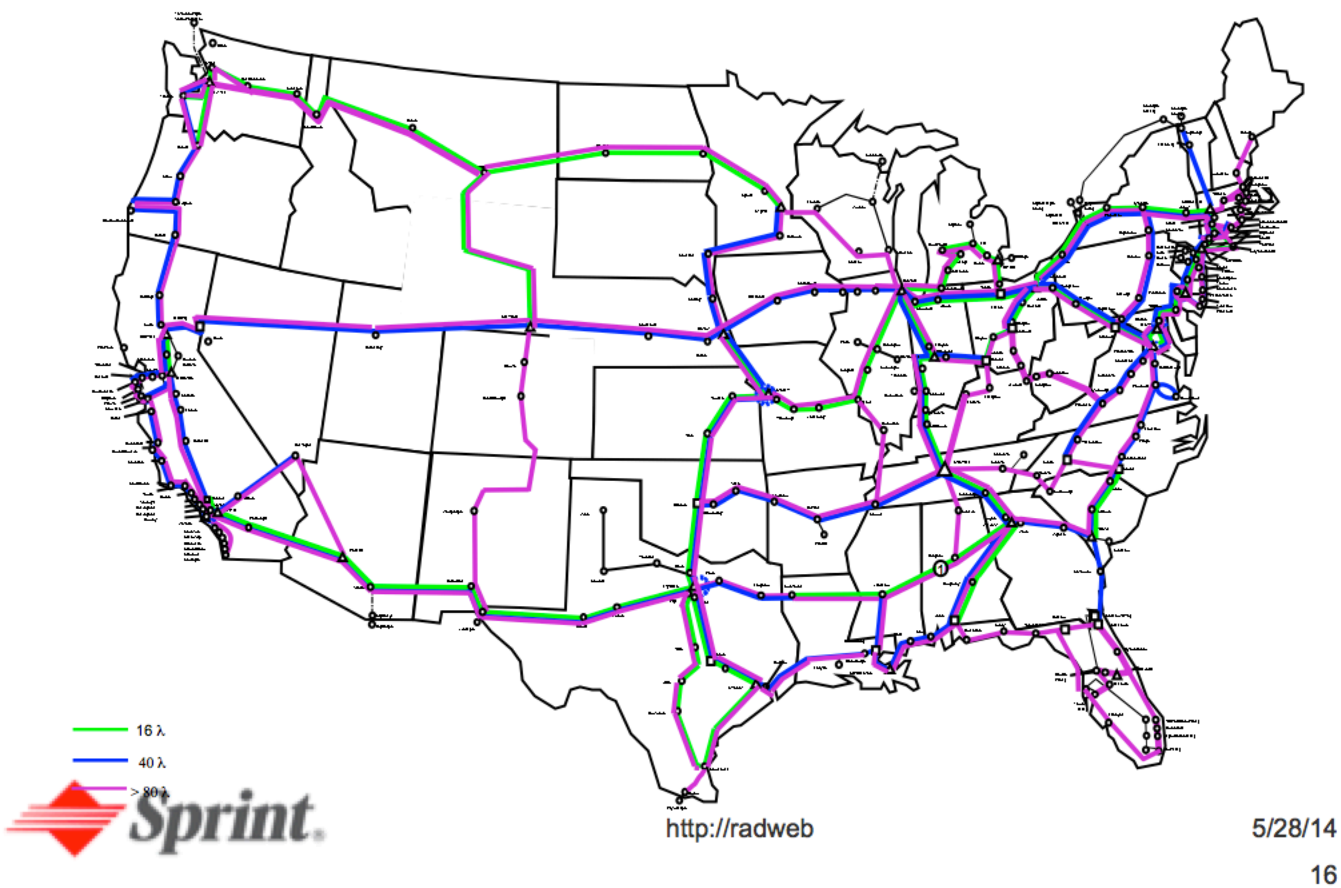
Danger Will Robinson!!!

Standard Disclaimer

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

So How Did I Get Involved In All Of This Complexity Stuff?

It all started here, circa 2000



What Happened?

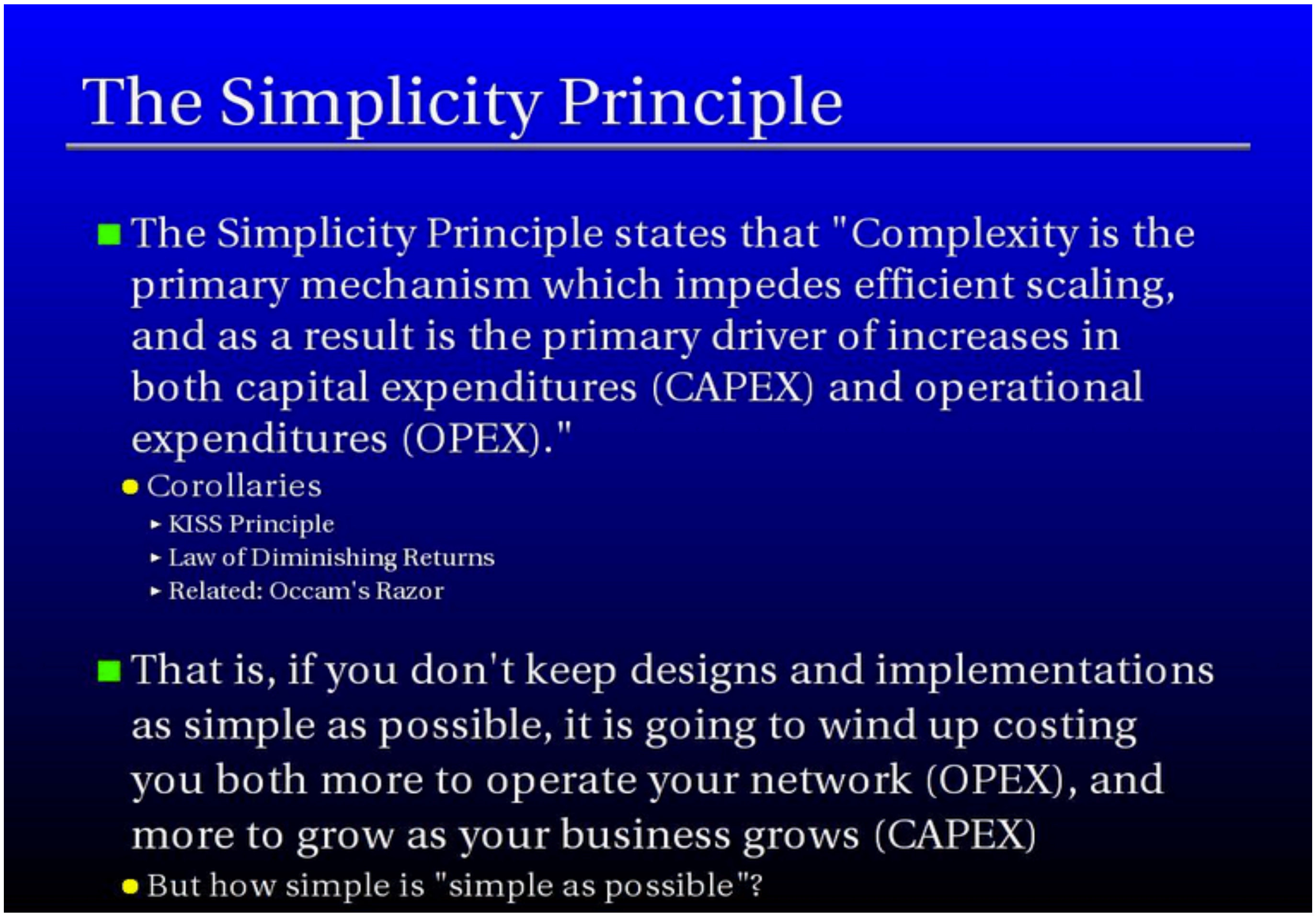
- Somewhere around 2000 I was re-org'ed to report up through Kansas City
  - along with all of the Sprintlink folks
  - Kansas City was famously the home of Pin Drop, ATM, Frame Relay, TDM, and the like

- We had been talking about how IP was so much "simpler" that FR/ATM/TDM/.. Predictably, first question I was asked by the KC folks was:
  - ***If the IP network is so much simpler, why is its OPEX/CAPEX profile so much higher (than ATM, FR, or TDM)?***

- **I could not answer this question**
  - **Seriously, I had no clue**
  - **There was all kinds of talk about FCAPS, NMS, etc, but none of it was helpful/quantitative**

- So I set out to understand how I might answer it
  - First by surveying the "complexity" literature
  - And BTW, what was complexity?
  - And surely there was a quantitative way to compare circuit and packet switched networks
    - **…or not**

One Result of this Exploration was RFC 3439
(Some Internet Architectural Guidelines and Philosophy)

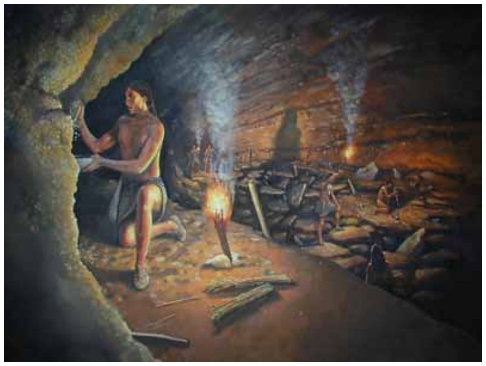


Mike O'Dell, by all accounts. See http://www.1-4-5.net/~dmm/talks/NANOG26/complexity_panel/

All Cool But Still, What is *Complexity*?

- Well, the "Simplicity Principle" *didn't tell us what complexity is or where it comes from*
  - We thought it had to do with *coupling* and *amplification*
  - The "SP" itself had no explanatory or predictive power
- Worse: Simplicity Principle approach contained a classic error
  - Confused symptoms (amplification, coupling) with root cause
  - Not to mention there was *no* theoretical/mathematical framework that we could lean on
- Result: over the past 12+ years I've been working with folks in the Control Theory, Systems Biology, and Engineering communities to try to get at this question
  - and understand its practical implications for engineers (us)

So I Started Looking At
The History of (Building) Construction
(why? Fred Harris)




Engineering Heuristics | Heuristic Materials "Science" | Mathematical Models

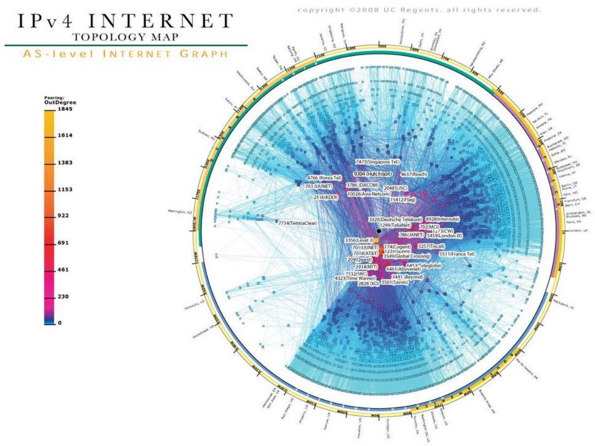
$$\tau = \|\mathbf{r}\| \, \|\mathbf{F}\| \sin\theta$$

Heuristic Knowledge

Engineering Heuristics | Models

Basic idea: *Engineering heuristics* take us a long (long) way, but if we want to scale beyond a certain point (say, build a 1000m tall building) we need a model of the building so we predict its behavior (likely including some knowledge of physics)

Applied to Scaling the Internet?



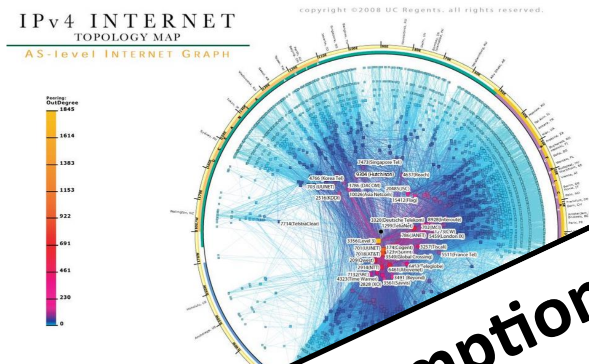
2^32 → 2^128

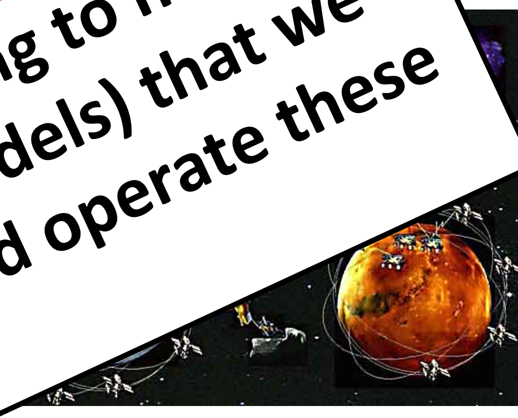
Engineering Heuristics
(sort of)

????

96 binary orders of magnitude
(ok, call it half)

Carrying on the Analogy…





The basic assumption here is that *to get to far greater levels of scale and functionality* we're going to need some kind of theoretical framework (models) that we can use to understand, design, build and operate these networks.

2^128

Models

$$
\begin{aligned}
\text{maximize} \quad & \sum_s U_s(x_s, P_{e,s}) + \sum_j V_j(w_j) \\
\text{subject to} \quad & \mathbf{R}\mathbf{x} \le \mathbf{c}(\mathbf{w}, \mathbf{P}_e), \\
& \mathbf{x} \in \mathcal{C}_1(\mathbf{P}_e), \quad \mathbf{x} \in \mathcal{C}_2(\mathbf{F}) \ \text{ or } \in \mathbf{\Pi}(\mathbf{w}), \\
& \mathbf{R} \in \mathcal{R}, \quad \mathbf{F} \in \mathcal{F}, \quad \mathbf{w} \in \mathcal{W}.
\end{aligned}
$$

Graphic courtesy CAIDA

So Goals for this Talk

- Characterize the essential features of *complexity*
  - and where find complexity both technological and biological systems
- Examine fundamental *tradeoffs*
  - that are made in complex systems
- Explore *universal architectural features*
  - and how they are related to tradeoffs and complexity
- Describe the relationship *between complexity, tradeoffs, and layering*
  - and how they can be part of a ***useful*** theoretical framework
- *Begin* to *Bridge* the Engineering and Theory Networking Communities
  - Theorists need to know what engineers known (what is real[1]), and
  - Engineers need the tools that we can get from theorists…

1 "Engineers always know first" – John Doyle

Said Another Way…

The major goal of this talk is to open up our thinking about what the essential architectural features of our network are, how these features combine to provide robustness (and its dual, fragility), and how the universal architectural features that we find in both technological and biological networks effect Internet robustness, scalability and evolvability.

Agenda

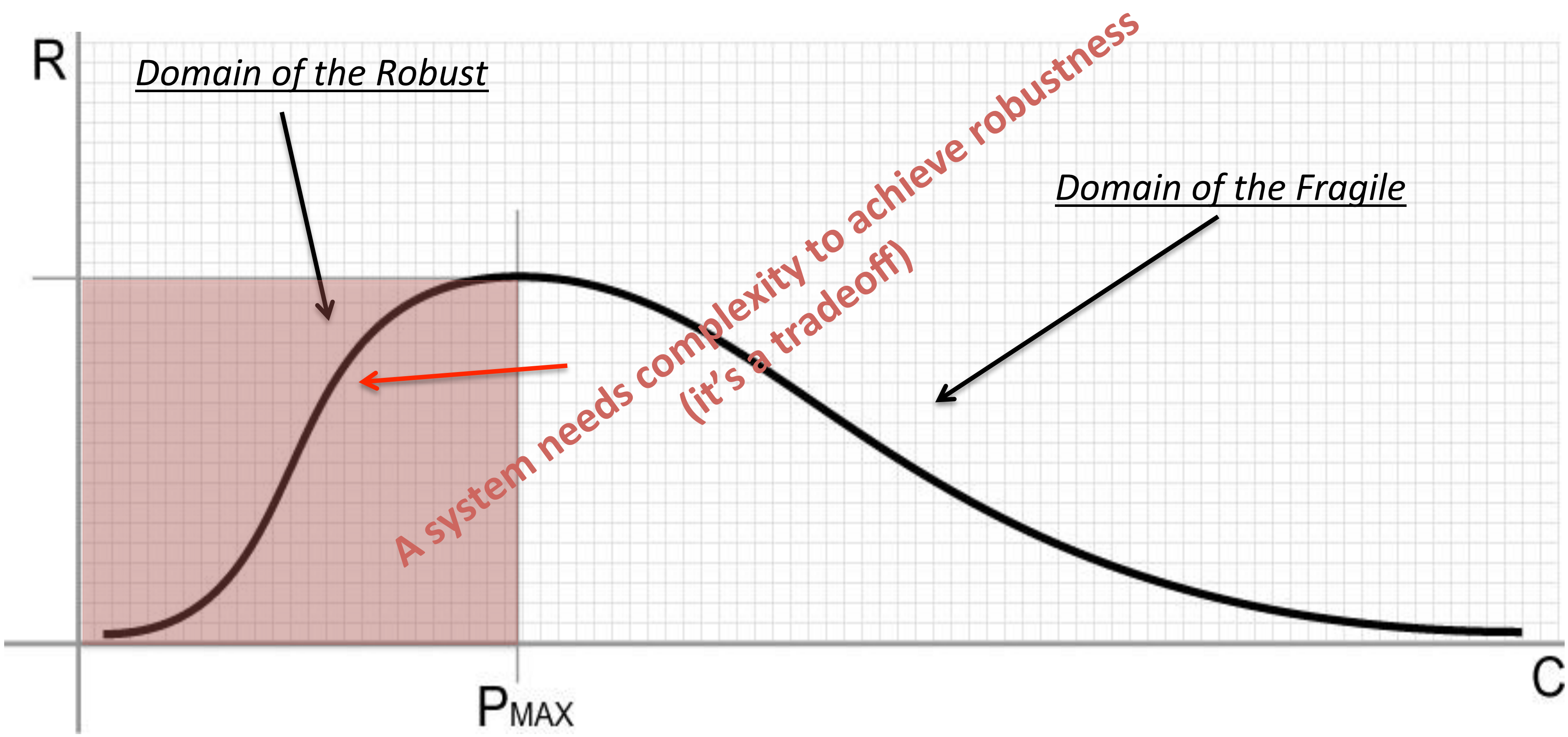
- ~~Too many words, too many slides ☺~~
  - ~~This talk is about thinking about networking in new ways~~
- ~~Motivation and Goals for this Talk~~
- What is Complexity and Why is it Hidden?
  - Robustness, Fragility, and Complexity
- The Architecture of Complex Systems
  - Universal Architectural Principles
- A Few Conclusions and Q&A if we have time

Ok, what is Complexity?

(and why is it hidden)

- Complexity is (mostly) ***hidden structure*** that arises in systems
- Its purpose is to create ***robustness*** to environmental and *component* ***uncertainty***
- Hidden?
  - Anti-lock/anti-skid brakes, packet loss (TCP), linux kernel, power grids, cloud stacks, SDN controllers, lunar landing systems, …
  - You don't notice they are there…until they fail
    - often catastrophically
- Why Hidden?
  - the hidden nature of complexity is a fundamental property of these systems
  - derives from universal architectural principles of complex systems (layering, constraints that deconstrain)
  - and is required to make robustness and evolvablity compatible
- But isn't complexity evil?
  - and we'll get to what robustness is in a sec…

Complexity Isn't Inherently "Evil"



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

So What Then is Robustness?

Robustness is a Generalized Feature of Complex Systems

- **Scalability** is *robustness* to changes to the size and complexity of a system as a whole
- **Reliability** is *robustness* to component failures
- **Efficiency** is *robustness* to resource scarcity
- **Modularity** is *robustness* to component rearrangements
- …

- So robustness is a very general idea
  – and captures many of the features we're seeking from the network

A Bit More Formally

- ***Robustness*** *is the preservation of a certain property in the presence of uncertainty in components or the environment*
  - Obviously a core Internet design principle
  - Systems Biology: Biological systems are designed such that 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
    - Exact adaptation in bacteria chemotaxis

- ***Fragility*** is the opposite of robustness
  - Another way to think about fragility
    - Technical: You are fragile if you depend on 2nd order effects (acceleration) and the "harm" curve is concave
  - A little more on this in the next few slides…

- 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)

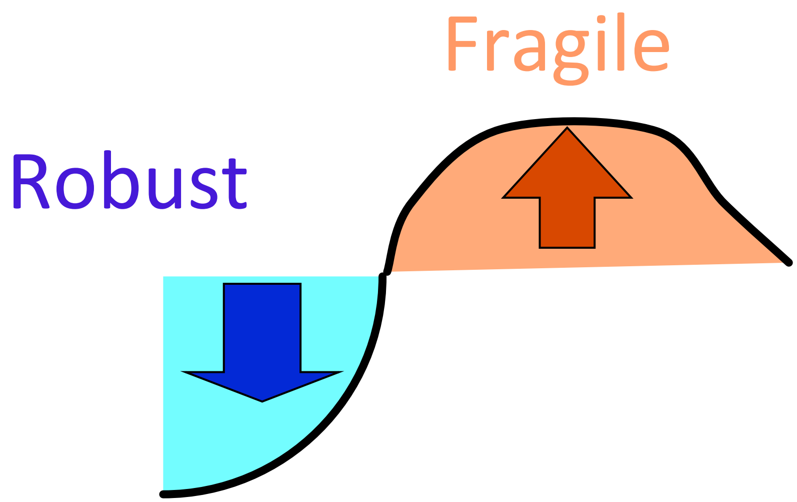
- For example, a possible ***RYF tradeoff*** is that a system with high efficiency (i.e., using minimal system resources) might be unreliable (i.e., fragile to component failure) or hard to evolve
  - VRRP , ISSU, HA, TE, {5,6,7..}-nines, …
  - Complexity/Robustness Spirals

Robust Yet Fragile?

(seems like a contradiction)

[a system] can have
[a property] that is ***robust*** to
[a set of perturbations]

Yet be ***fragile*** for
[a different property]
Or [a different perturbation]



Recent results suggest that the RYF tradeoff is a hard tradeoff that cannot be overcome[1]

This is profound: If you create robustness somewhere you will create fragility somewhere else…OK, but where?

Network Engineering, along with most other engineering disciplines, does not explicitly (or otherwise) account for this effect

Harm Function: Concave → Fragile, Convex → Robust

[1] See Marie E. Csete and John C. Doyle, "Reverse Engineering of Biological Complexity", http://www.cds.caltech.edu/~doyle/wiki/images/0/05/ScienceOnlinePDF.pdf

Interestingly, Fragility and Scaling are Related

- 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***

- → *A big event hurts non-linearly more than the sum of small events*

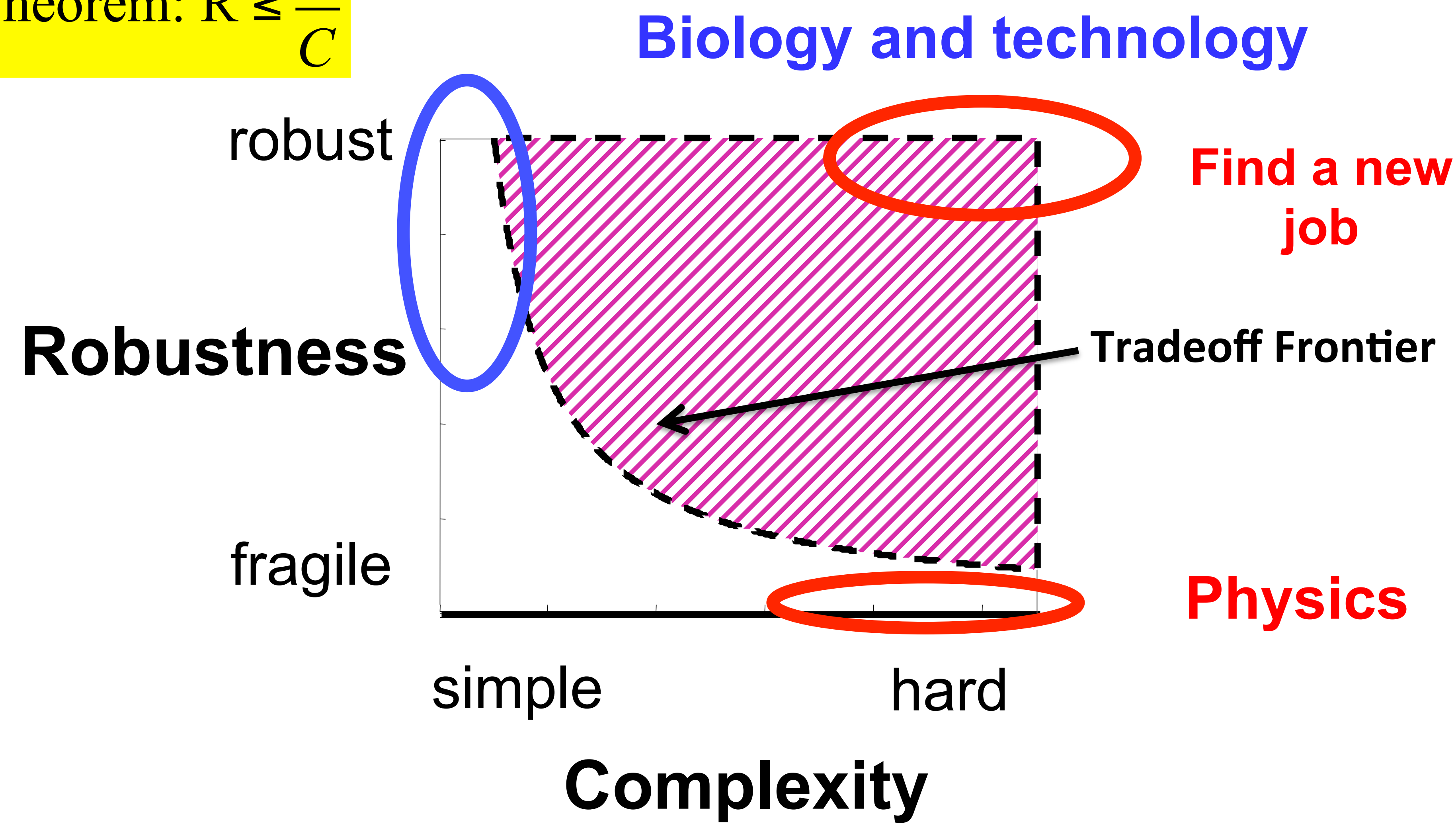
- For example, a coffee cup on a table suffers non-linearly more from large deviations (*H(p, nz)*) than from the cumulative effect of smaller events (*nH(p,z)*)
  - So the cup is damaged far more by *tail events* than those within a few σ's of the mean
  - Sensitivity to tail events → RYF
  - Too theoretical? 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

- So 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, 3 σ)
  - Again, ARP storms, congestion collapse, AS 7007, DDOS, … → non-linear damage

Coffee cup example courtesy Nassim Taleb. See http://www.fooledbyrandomness.com

So Its All About (RYF) Tradeoffs

Theorem: $R \leq \frac{1}{C}$



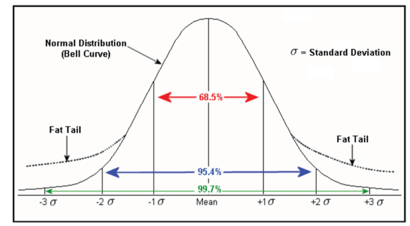
BTW, RYF Behavior is Everywhere

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. That is, they are RYF-Complex
- Often involving hijacking/predation/exploitation of the same mechanism
  - We've certainly seen this in the Internet space (consider DDOS of various varieties)
- These are hard constraints (RYF behavior is conserved)

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*
    - Look ahead: Layering, Bowties/Hourglasses, Constraints that Deconstrain
  - Managing spiraling complexity/fragility
  - Not predicting what is likely or typical
    - But rather understanding what is catastrophic
    - or in Taleb's terminology, that which is *fat tailed*



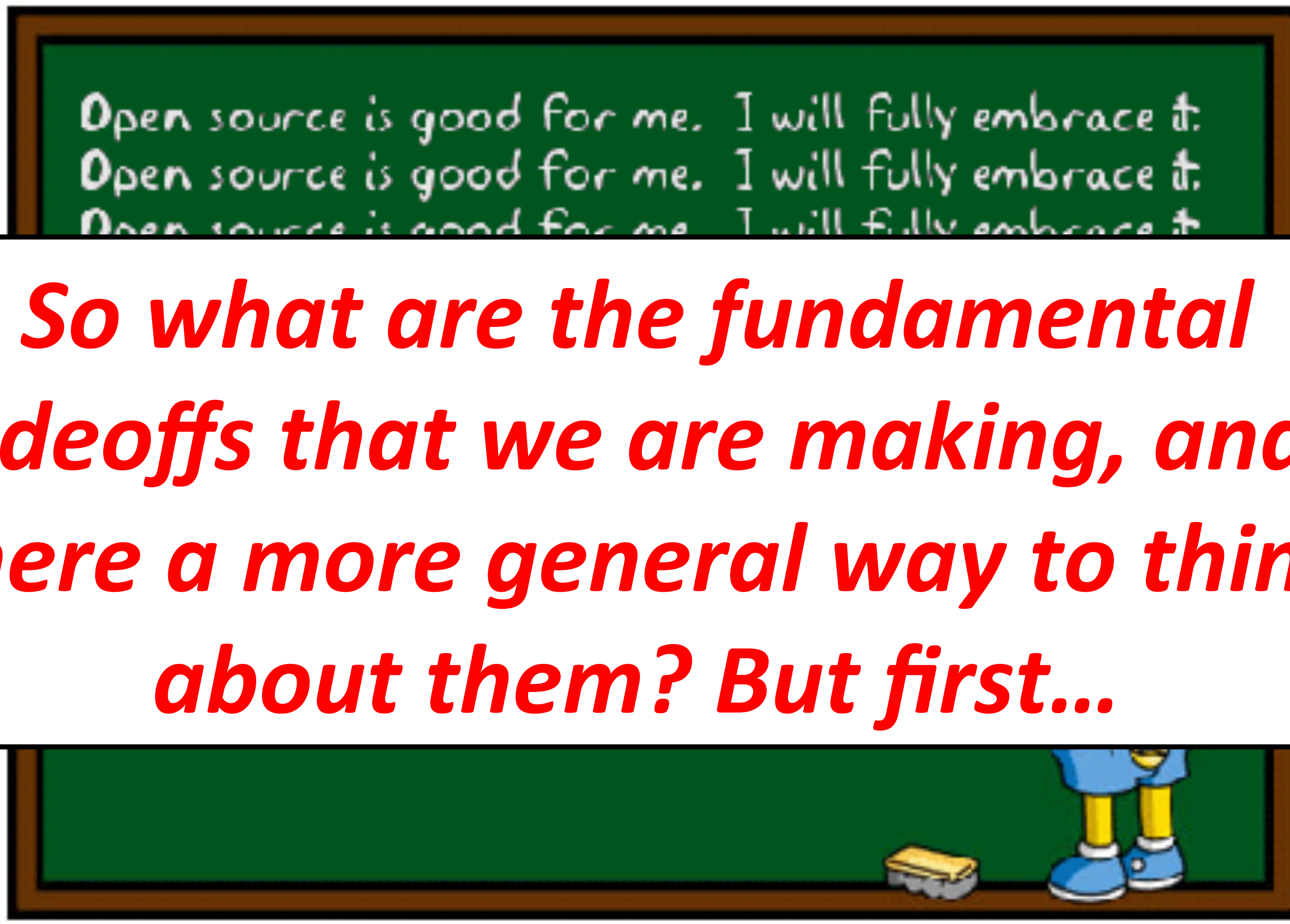
- And BTW, it is much easier to create the robust features than it is to prevent the fragilities
  - And as I mentioned, there are poorly understood "conservation laws" at work[1]

- Bottom Line
  - ***Understanding RYF behavior and associated tradeoffs means understanding network architecture and the hidden nature of complexity***

[1] See Marie E. Csete and John C. Doyle, "Reverse Engineering of Biological Complexity", http://www.cds.caltech.edu/~doyle/wiki/images/0/05/ScienceOnlinePDF.pdf

BTW, What is our Archite ?
(and what tradeoffs are)

Grade,



So what are the fundamental tradeoffs that we are making, and is there a more general way to think about them? But first…

I E T F

What tradeoffs are embedded in what we do everyday?

```
00 00 00  00 00 00 00  03 00 00 00  00 00 00 00  ................................
00 00 00  14 00 00 00  00 00 00 00  07 00 00 00  ..`............H.................
00 00 00  00 00 00 00  A0 03 40 00  00 00 00 00  .............@.............@.....
00 00 00  18 00 00 00  00 00 00 00  FE FF FF 6F  ...............................o
00 00 00  00 00 00 00  F0 FF FF 6F  00 00 00 00  ......@........o.............o....
00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  t.@..............................
00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................................
00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................................
00 00 00  00 00 00 00  36 04 40 00  00 00 00 00  ....P.`...................6.@.....
00 00 00  00 00 00 00  00 00 00 00  47 43 43 3A  F.@.....V.@...................GCC:
2D 31 75  62 75 6E 74  75 35 29 20  34 2E 36 2E   (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.
68 73 74  72 74 61 62  00 2E 69 6E  74 65 72 70  3...symtab..strtab..shstrtab..interp
67 6E 75  2E 62 75 69  6C 64 2D 69  64 00 2E 67  ..note.ABI-tag..note.gnu.build-id..g
74 72 00  2E 67 6E 75  2E 76 65 72  73 69 6F 6E  nu.hash..dynsym..dynstr..gnu.version
2E 64 79  6E 00 2E 72  65 6C 61 2E  70 6C 74 00  ..gnu.version_r..rela.dyn..rela.plt.
64 61 74  61 00 2E 65  68 5F 66 72  61 6D 65 5F  .init..text..fini..rodata..eh_frame_
2E 64 74  6F 72 73 00  2E 6A 63 72  00 2E 64 79  hdr..eh_frame..ctors..dtors..jcr..dy
64 61 74  61 00 2E 62  73 73 00 2E  63 6F 6D 6D  namic..got..got.plt..data..bss..comm
00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ent...............................
00 00 00  00 00 00 00  00 00 00 00  1B 00 00 00  ................................
```

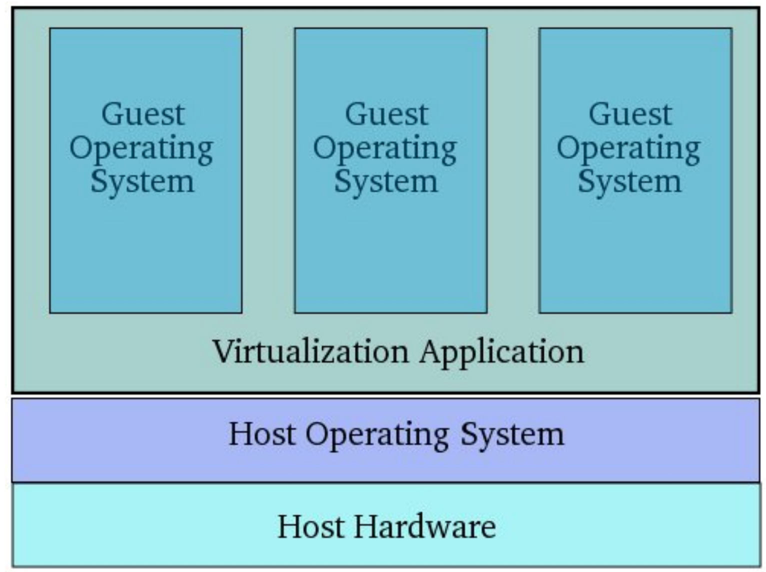
```python
def add5(x):
    return x+5

def dotwrite(ast):
    nodename = getNodename()
    label=symbol.sym_name.get(int(ast[0]),ast[0])
    print ' %s [label="%s' % (nodename, label),
    if isinstance(ast[1], str):
        if ast[1].strip():
            print '= %s"];' % ast[1]
        else:
            print '"]'
    else:
        print '"];'
        children = []
        for n, child in enumerate(ast[1:]):
            children.append(dotwrite(child))
        print ' %s -> {' % nodename,
        for name in children:
            print '%s' % name,
```

What tradeoffs are being made here?

Speed vs. flexibility?

How About Here?



Speed vs. flexibility (bare metal vs. VM vs. container)?

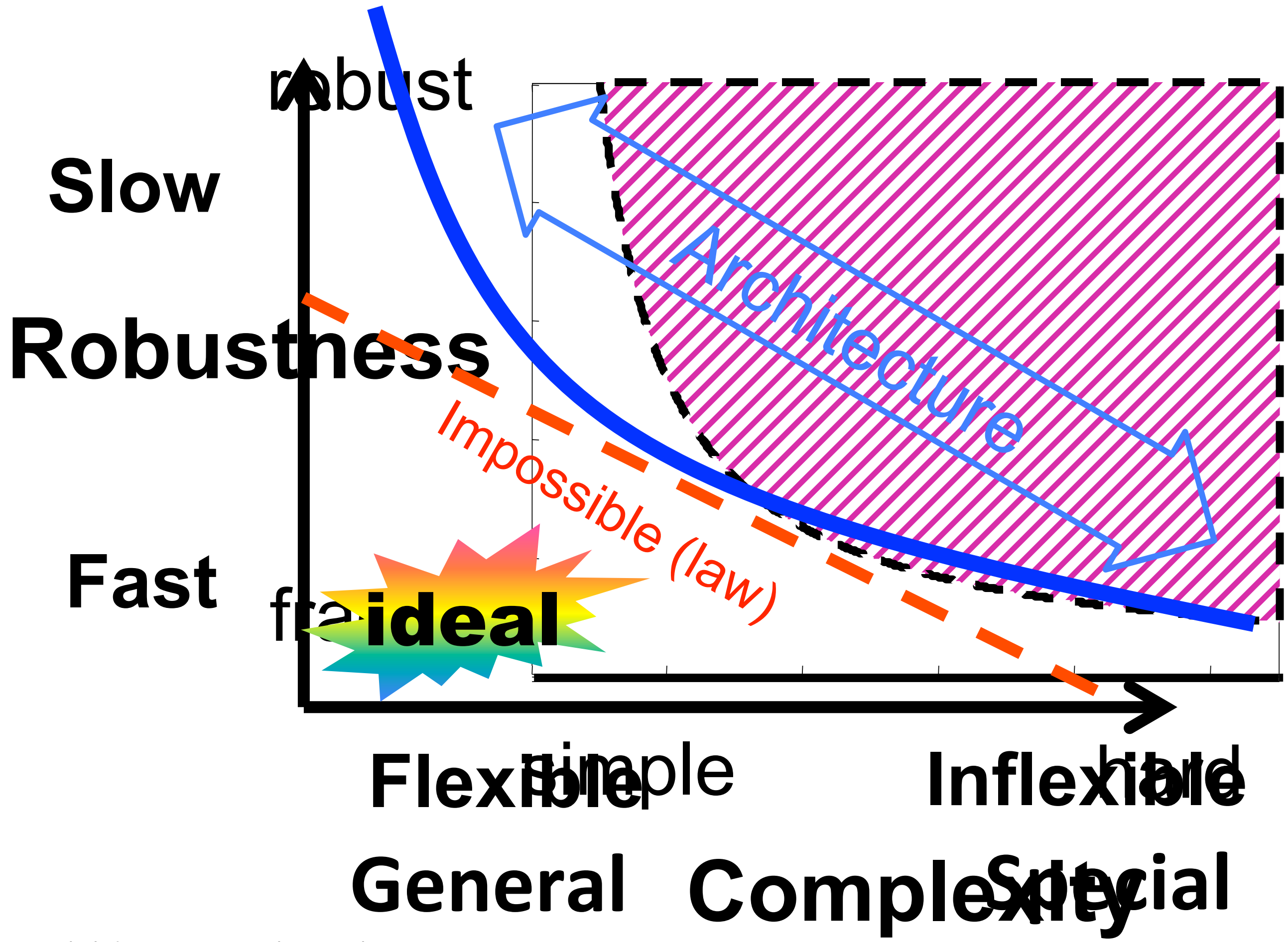
Summary: Common Tradeoffs

- Binary machine code vs. (interpreted) higher-level language
- VM vs. bare metal (vs. container)
- Fast path vs. slow path
- Hardware vs. software
- Convergence time vs. state
- …
- But what are the essential features of these tradeoffs?
  - What is fundamental in the tradeoff space?
    - And are there "laws" governing these tradeoffs?
    - And can we derive useful engineering laws from these "laws"?
  - And how do they relate to RYF-complexity?

Turns out that RYF tradeoffs are fundamental

Example: Computational Complexity

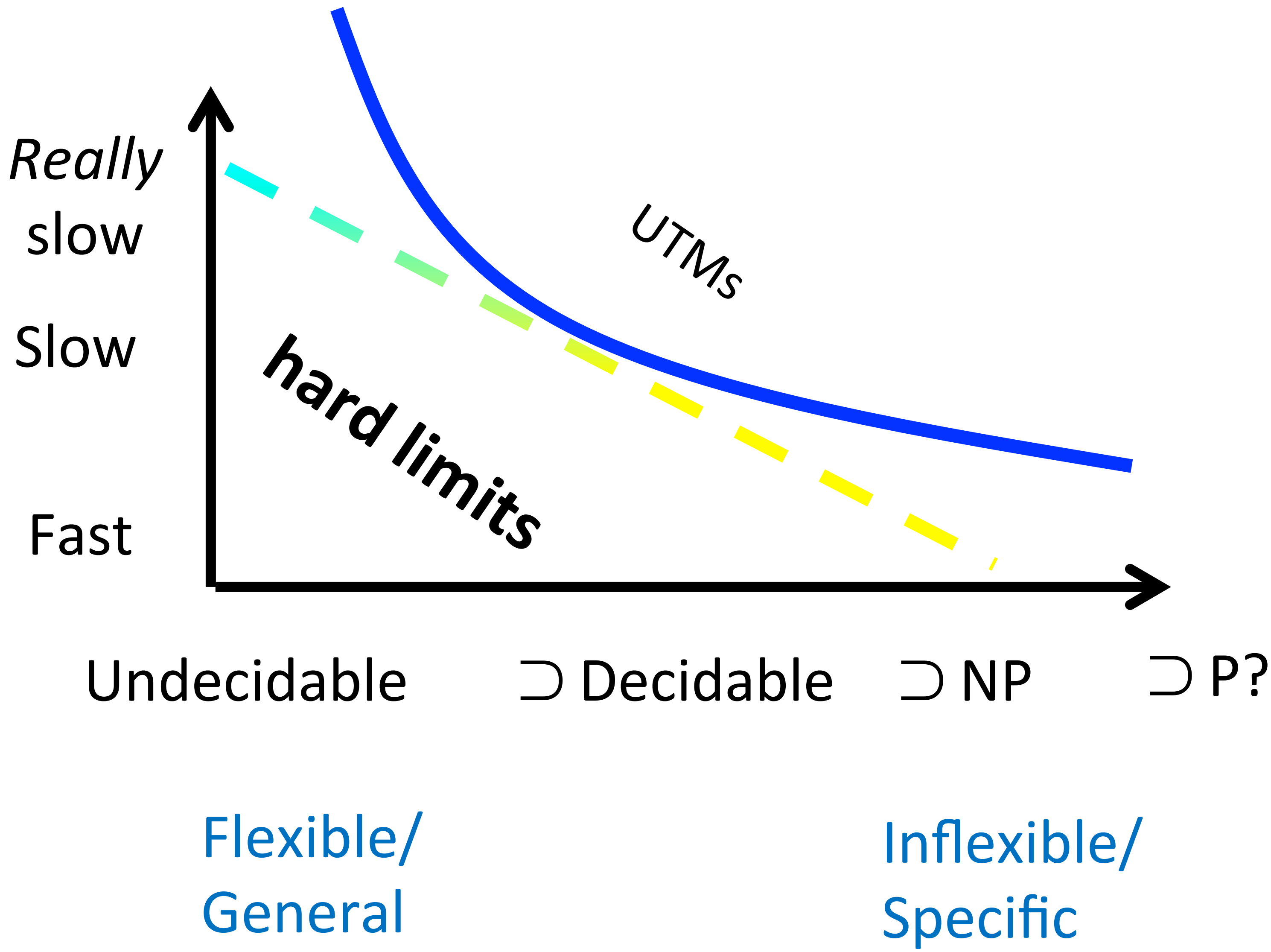
Layering, Formal Systems, Hard Tradeoffs



Theorem: $R \leq \frac{1}{C}$

Original slide courtesy John Doyle

Drilling down a bit on the Computational Complexity Tradeoff Space (changing axes)

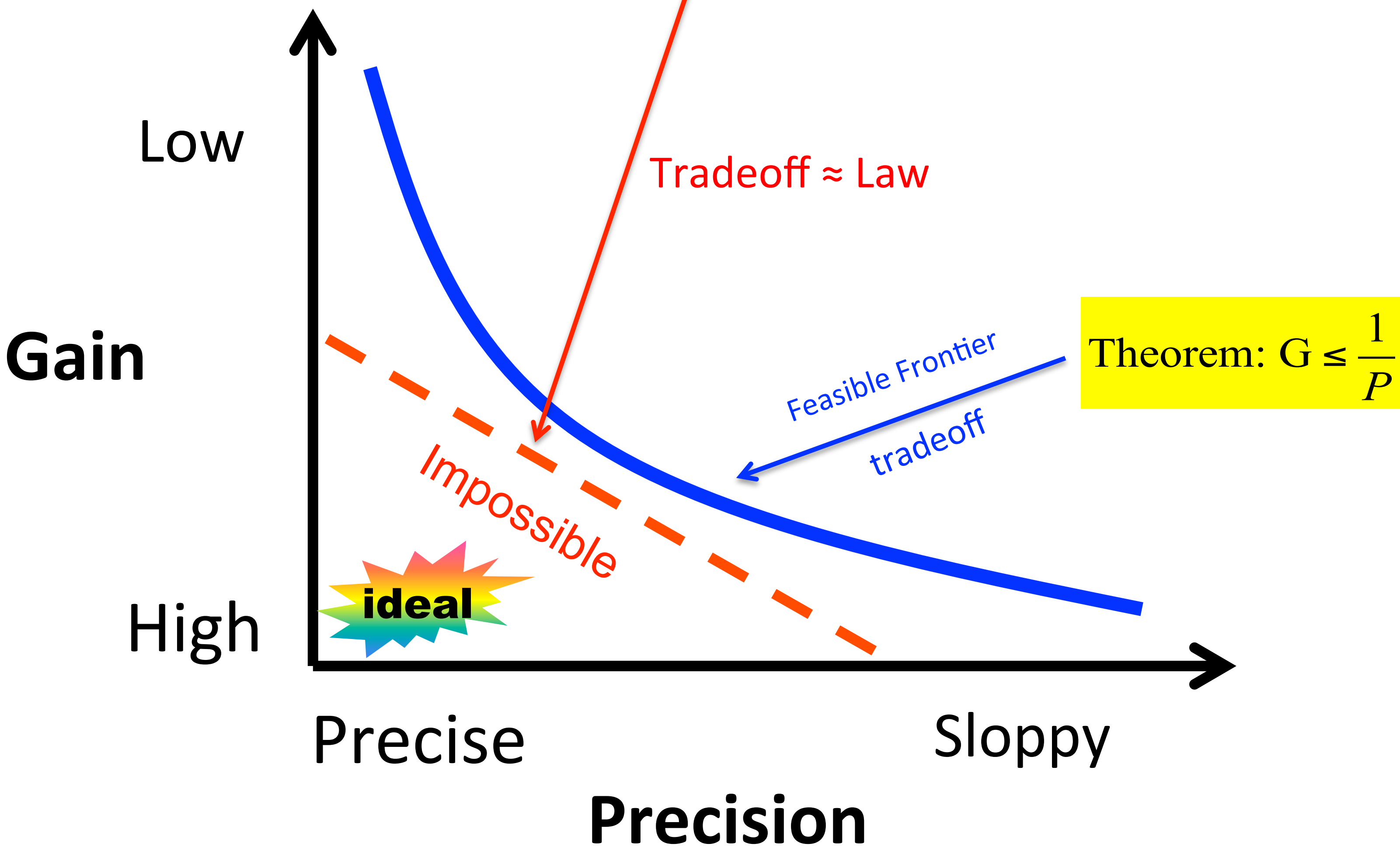


Another Example: Feedback Control Theory

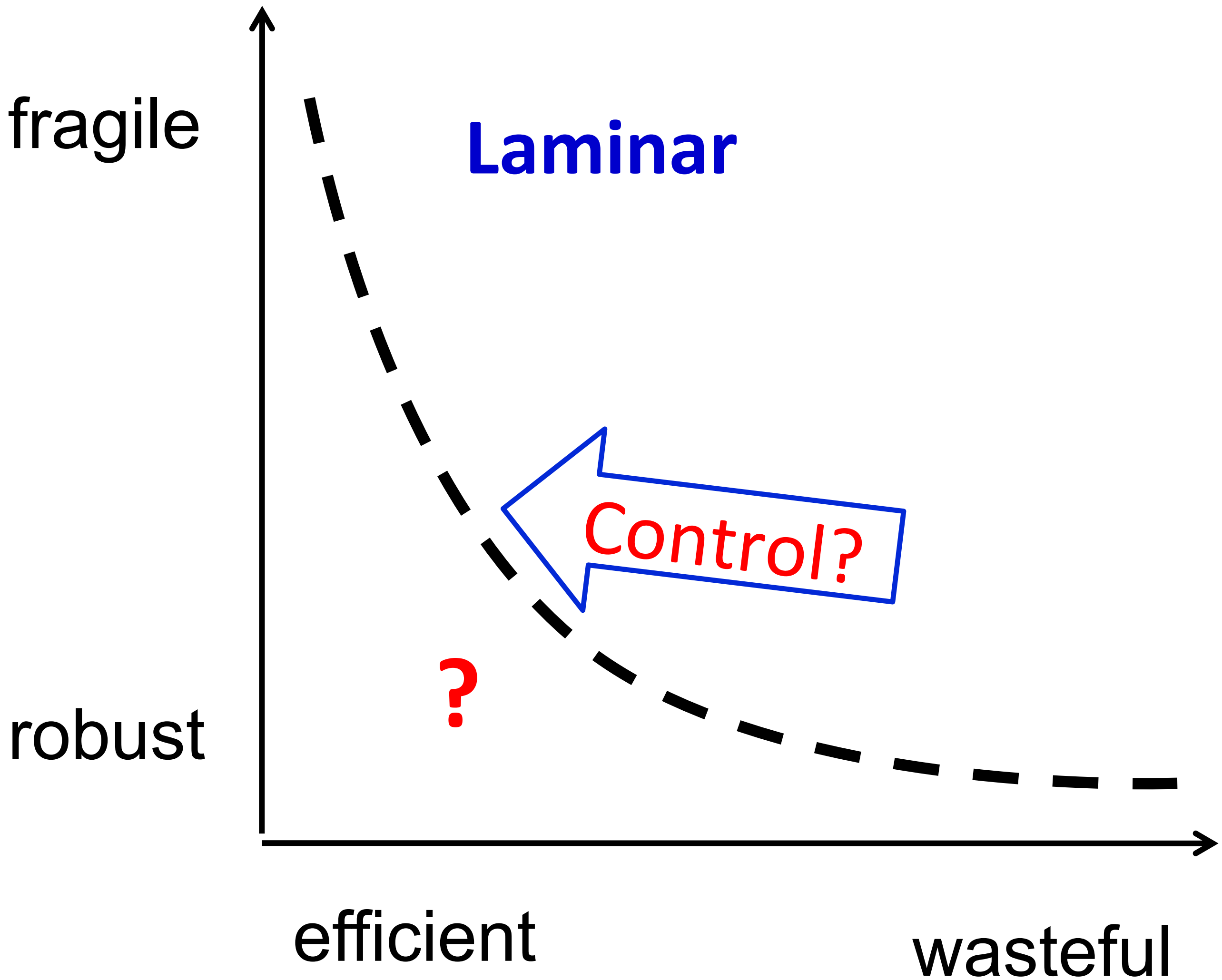
(Gain/Sensitivity Tradeoff In Feedback Control)

$$\int_0^\infty \ln|S(i\omega)|d\omega = \int_0^\infty \ln\left|\frac{1}{1+L(i\omega)}\right|d\omega = \pi\sum Re(p_k) - \frac{\pi}{2}\lim_{s\to\infty} sL(s)$$

Bode Sensitivity Integral

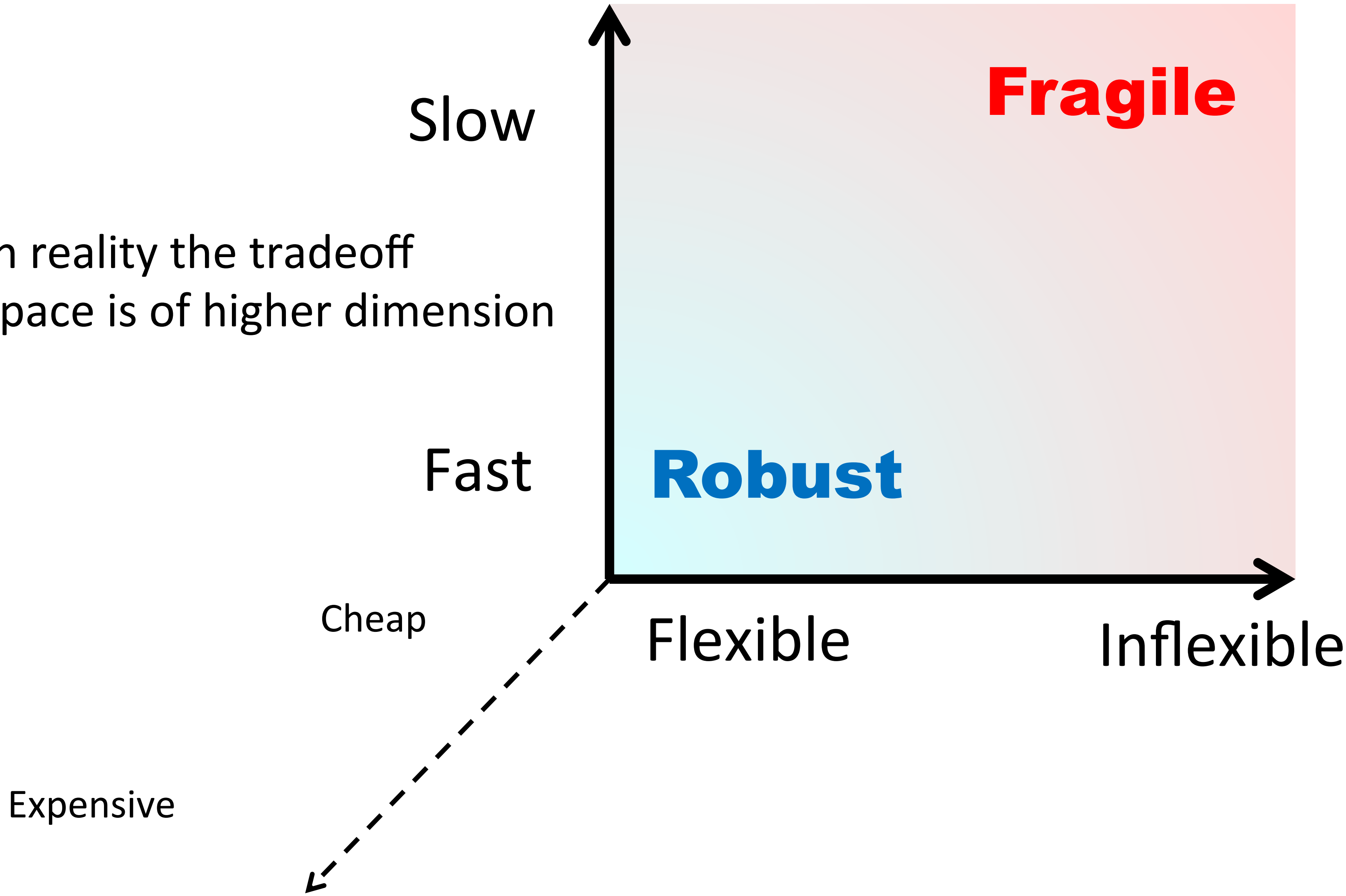


Example: Laminar Flow



Efficient vs Robust

In reality the tradeoff space is of higher dimension



Hopefully not too high dimension: *The Curse of Dimensionality*

Agenda

- ~~Too many words, too many slides ☺~~
  - ~~This talk is about thinking about networking in new ways~~

- ~~Motivation and Goals for this Talk~~

- ~~What is Complexity and Why is it Hidden~~
  - ~~Robustness, Fragility, and Complexity~~

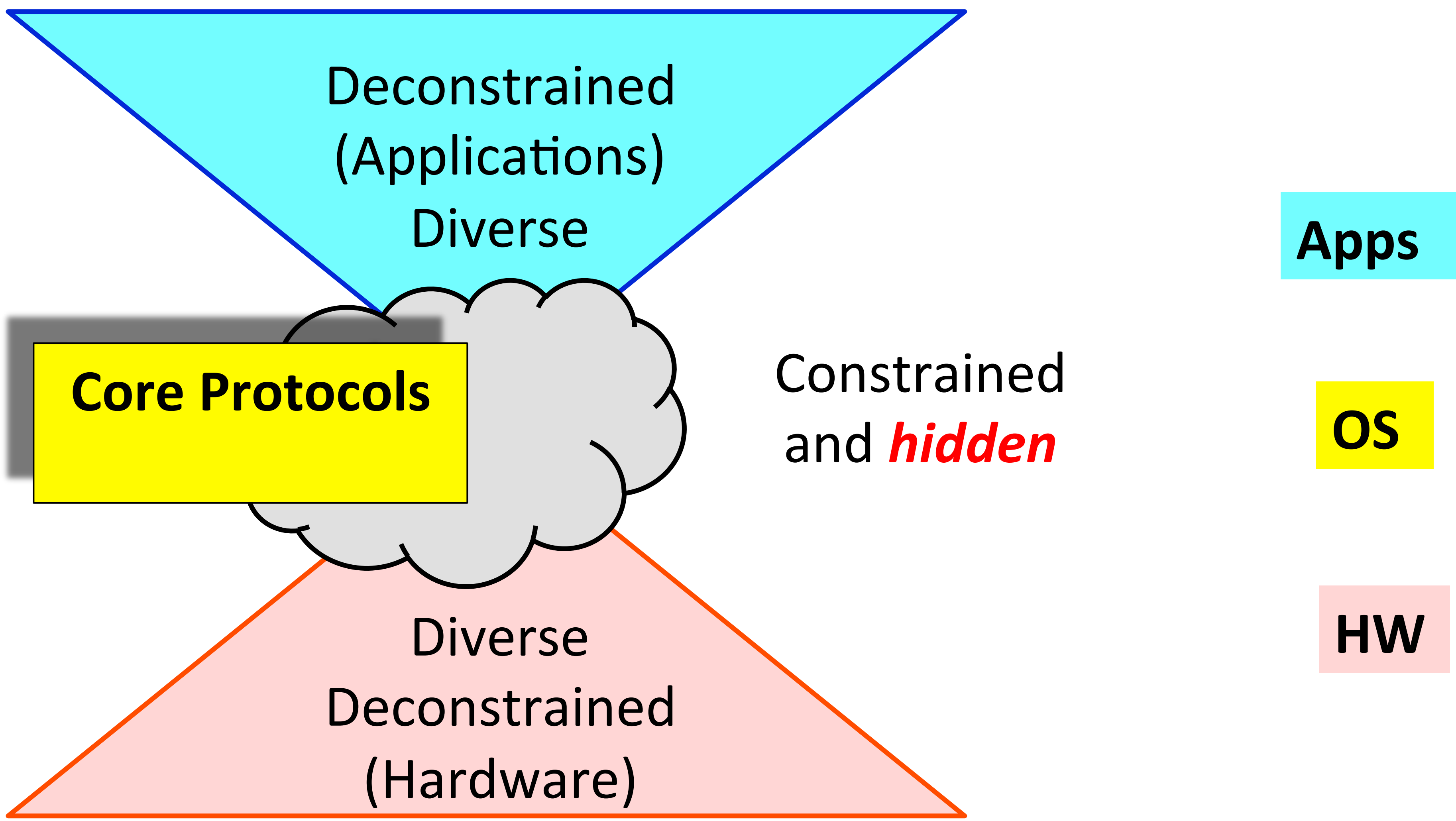
- The Architecture of Complex Systems
  - Universal Architectural Principles

- A Few Conclusions and Q&A if we have time

The Architecture of Complex Systems

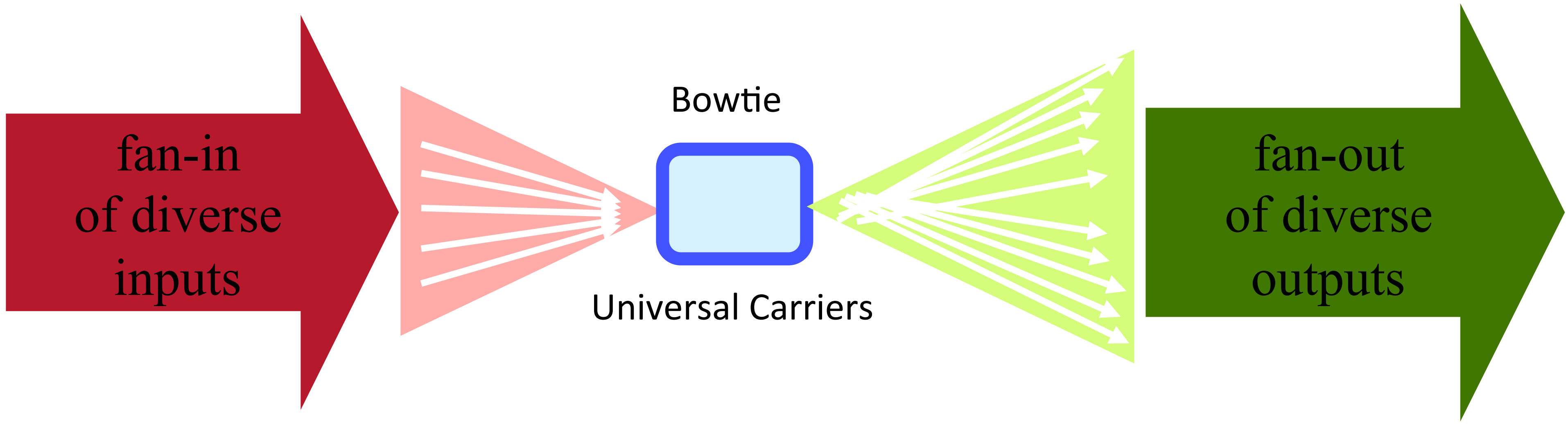
- What we have learned is that there are ***universal architectural building blocks*** found in systems that scale and are evolvable. These include
- **Architecture/Layering**
- **Laws, constraints, tradeoffs**
- Protocol Based Architectures
- Massively distributed with *robust* control loops
- Consequences
  - Hidden RYF Complexity
  - Hijacking, parasitism, predation

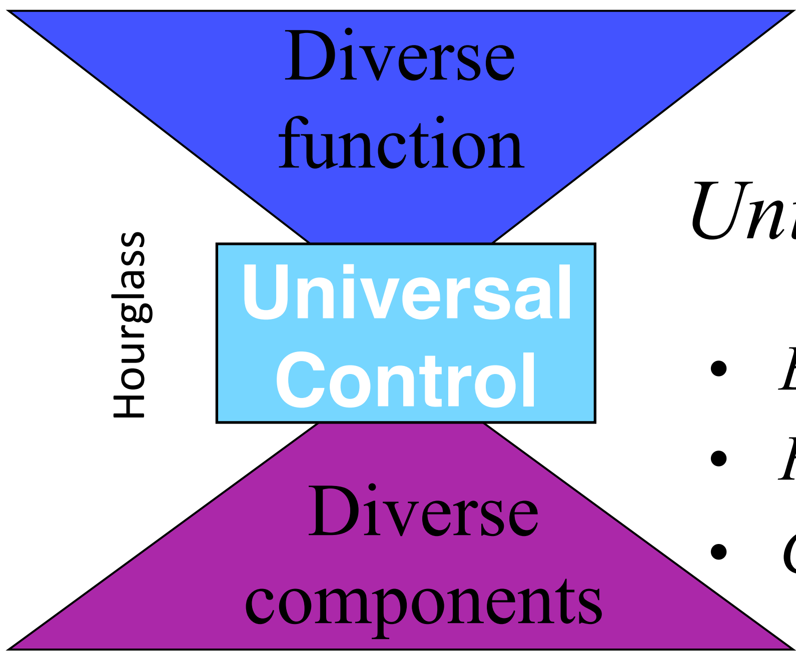
So What Do We Know About Architecture?
(What is the fundamental architecture of complex systems?)



Slide courtesy John Doyle

Bowties, Hourglasses and Layering



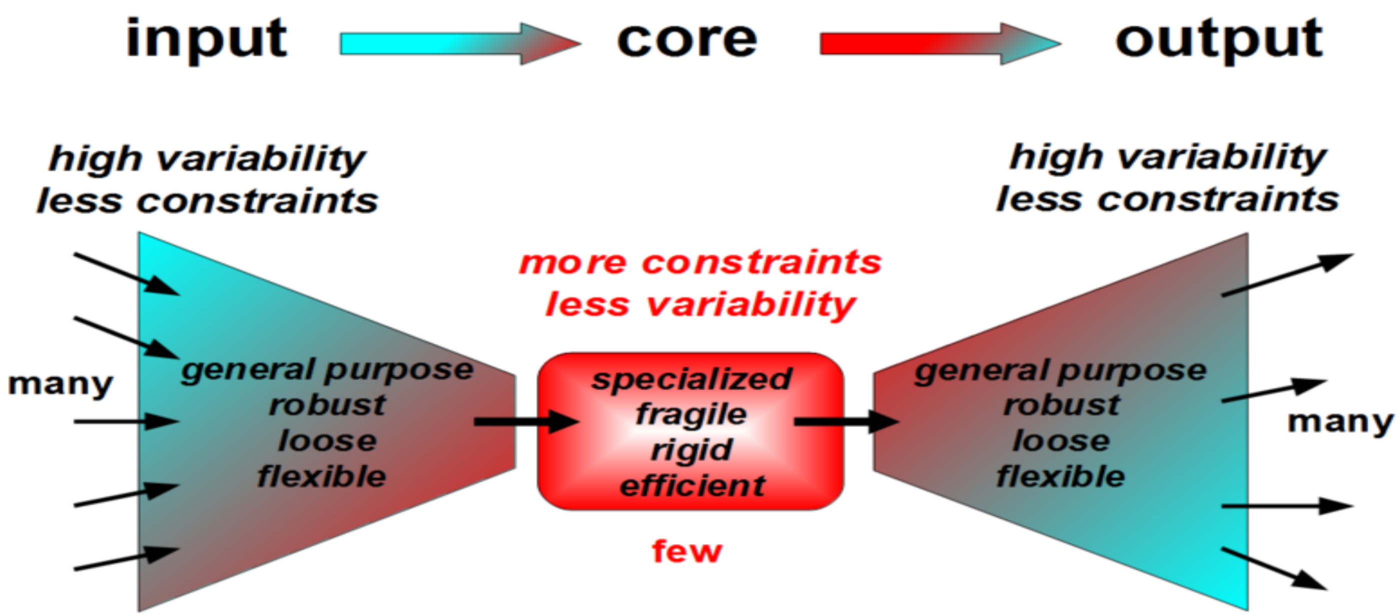


Universal Architectural Principles

- *Bowties* for *flows within layers* (protocol)
- *Hourglasses* for *layering of control* (stack)
- *Constraints that Deconstrain*

Bowties 101

Constraints that Deconstrain
Schematic of a "Layer"



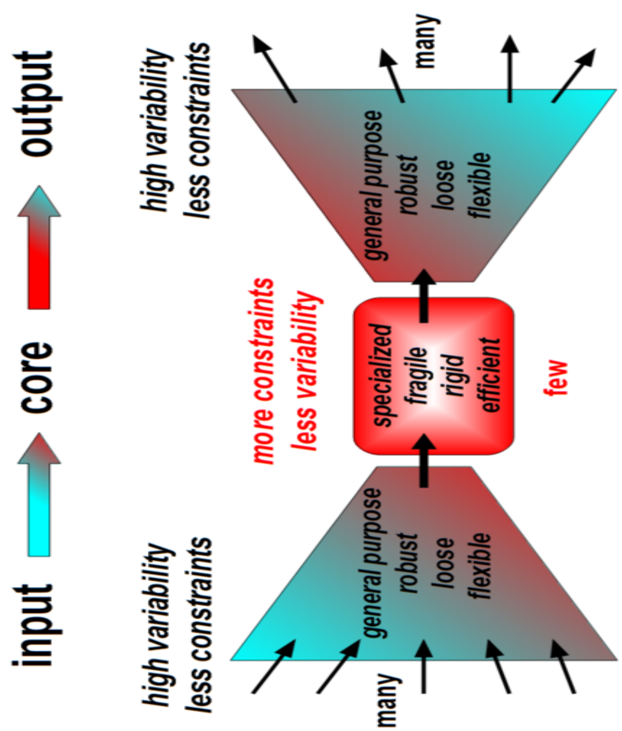
For example, the reactions and metabolites of core metabolism, e.g., Adenosine Triphosphate (*ATP) metabolism*, Krebs/Citric Acid Cycle, … form a "metabolic knot". That is, ATP is a ***Universal Carrier*** for cellular energy.

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

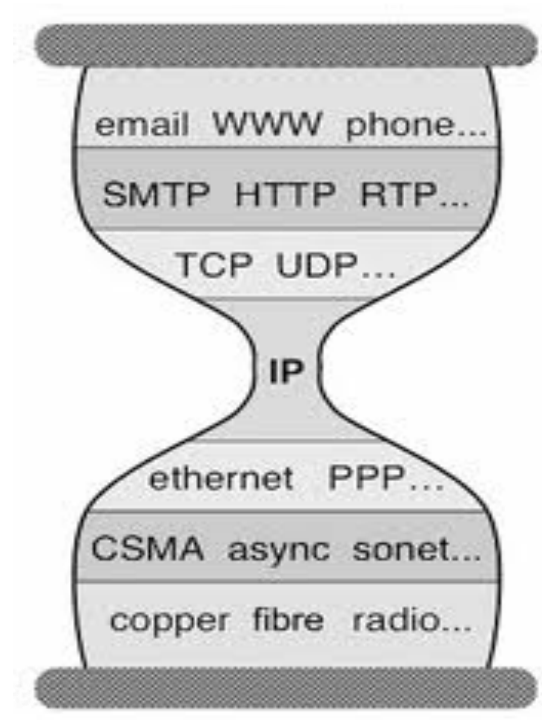
See Kirschner M., and Gerhart J., "Evolvability", Proc Natl Acad Sci USA , 95:8420–8427, 1998.

But Wait a Second

Anything Look Familiar?



Bowtie Architecture

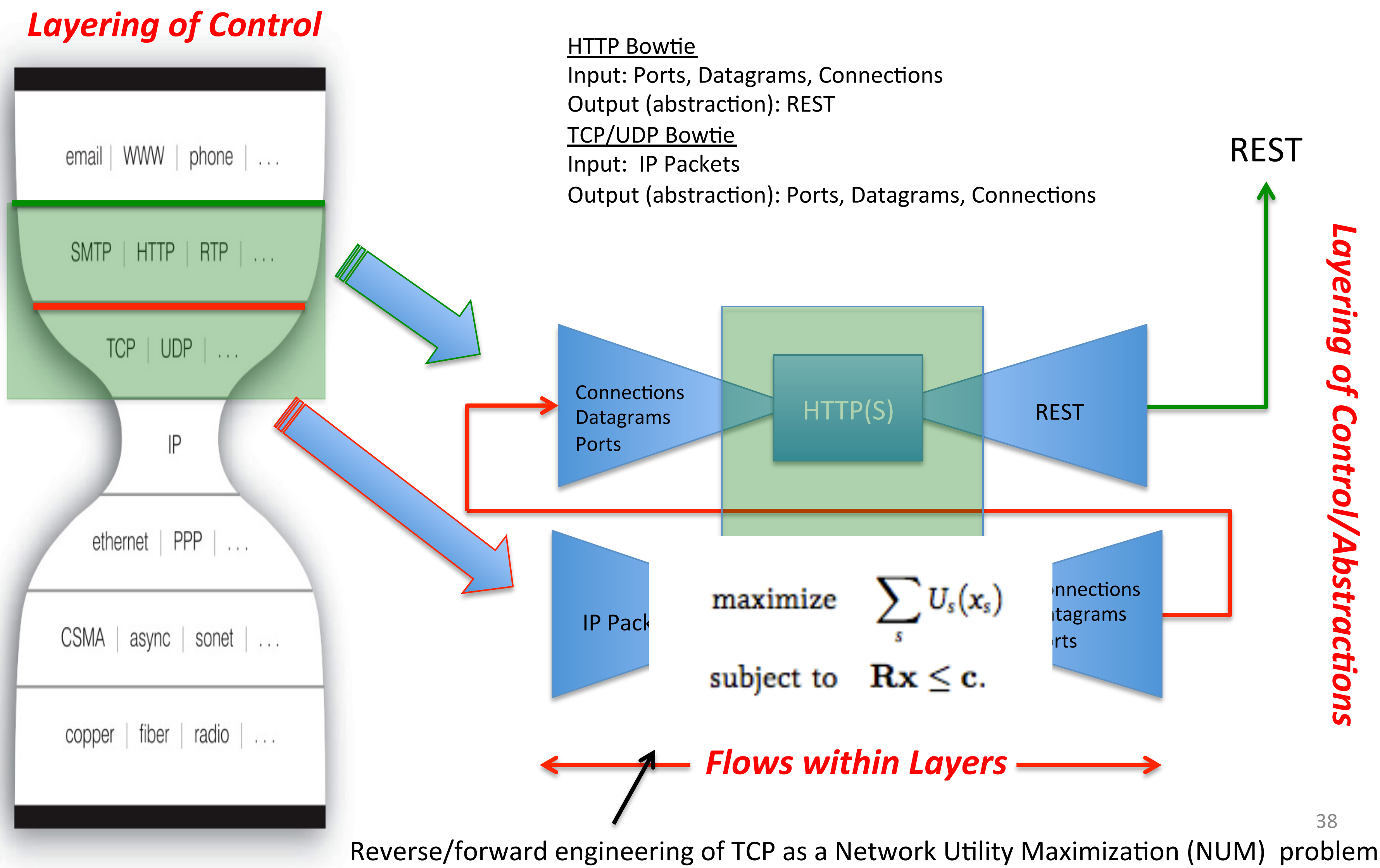


Hourglass Architecture

Comes down to whether you see layering as horizontal or vertical

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.

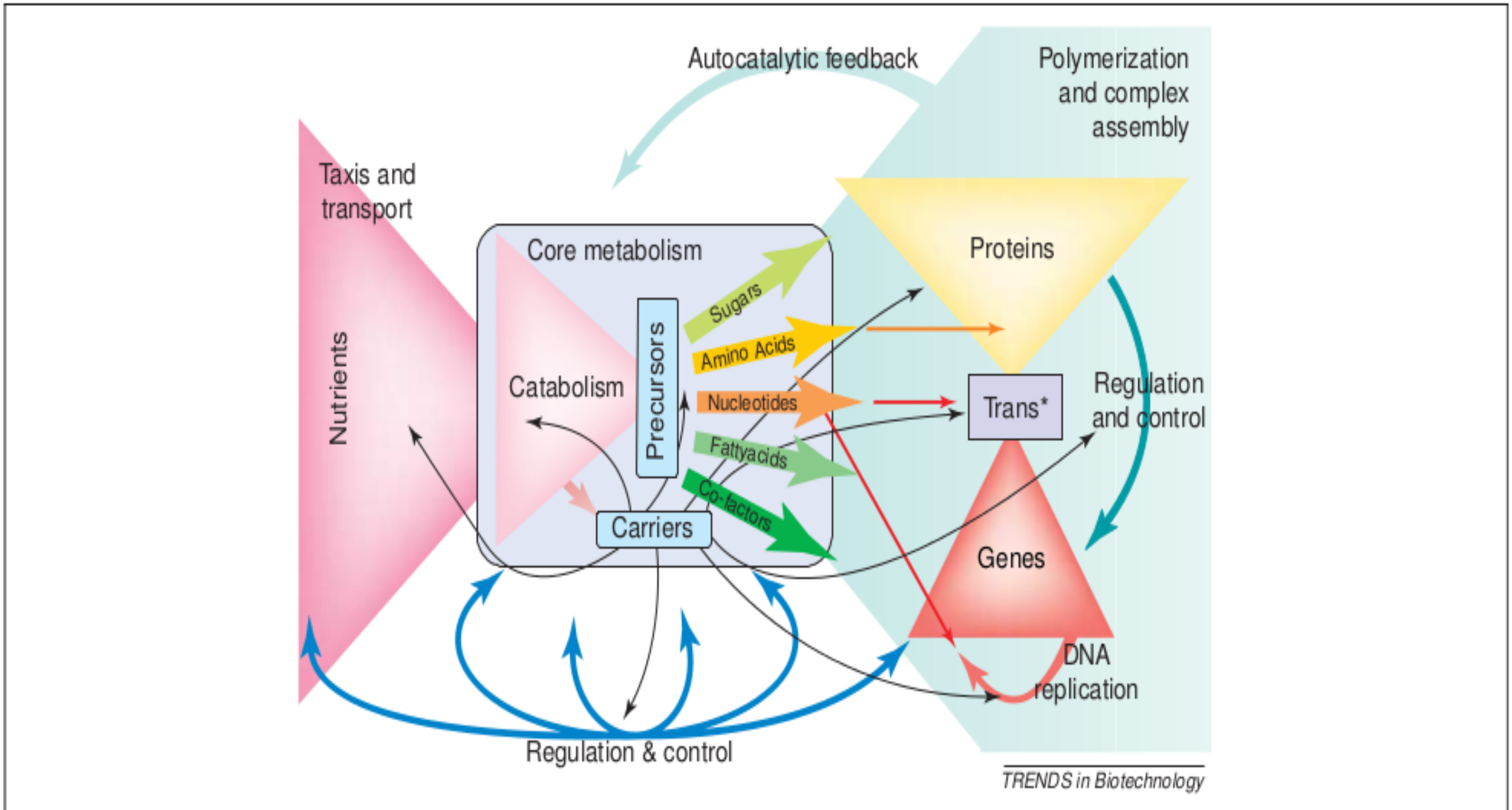
The Nested Bowtie/Hourglass Architecture of the Internet



Reverse/forward engineering of TCP as a Network Utility Maximization (NUM) problem

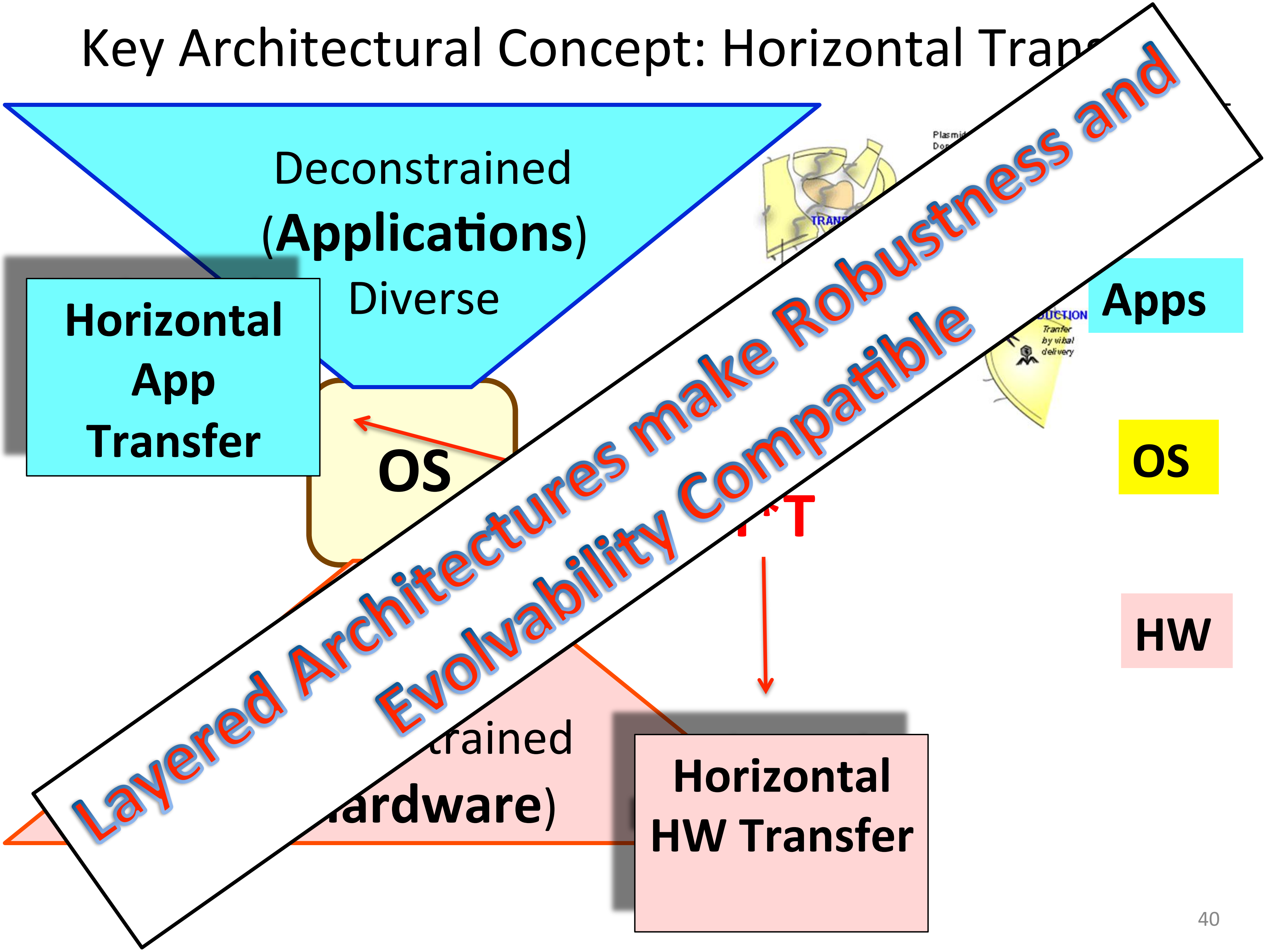
In Practice Things are More Complicated

The Nested Bowtie/Hourglass Architecture of Metabolism

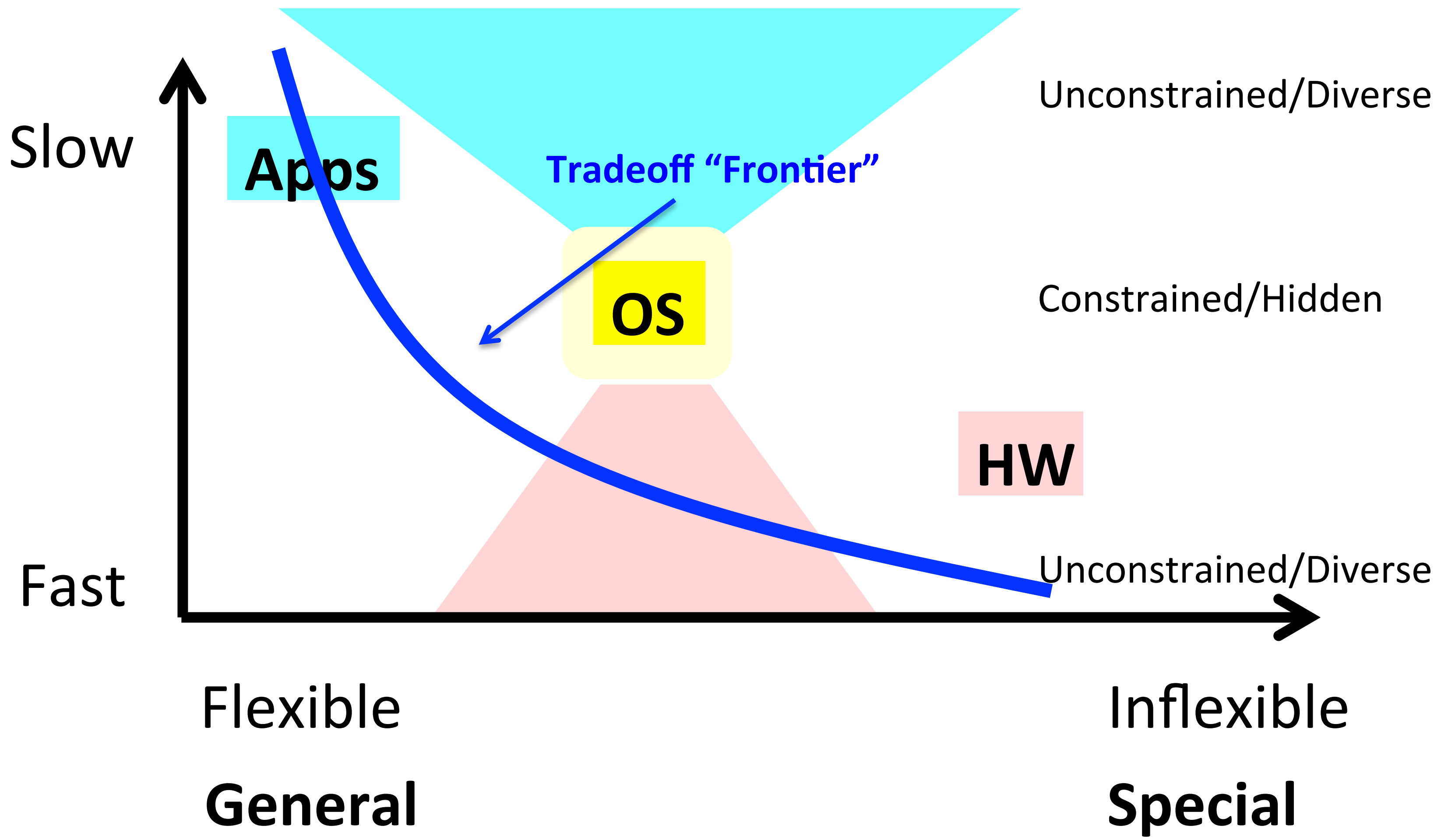


See Csete, M. and J. Doyle, “Bowties, metabolism and disease”, TRENDS in Biotechnology, Vol. 22 No. 9, Sept 2004

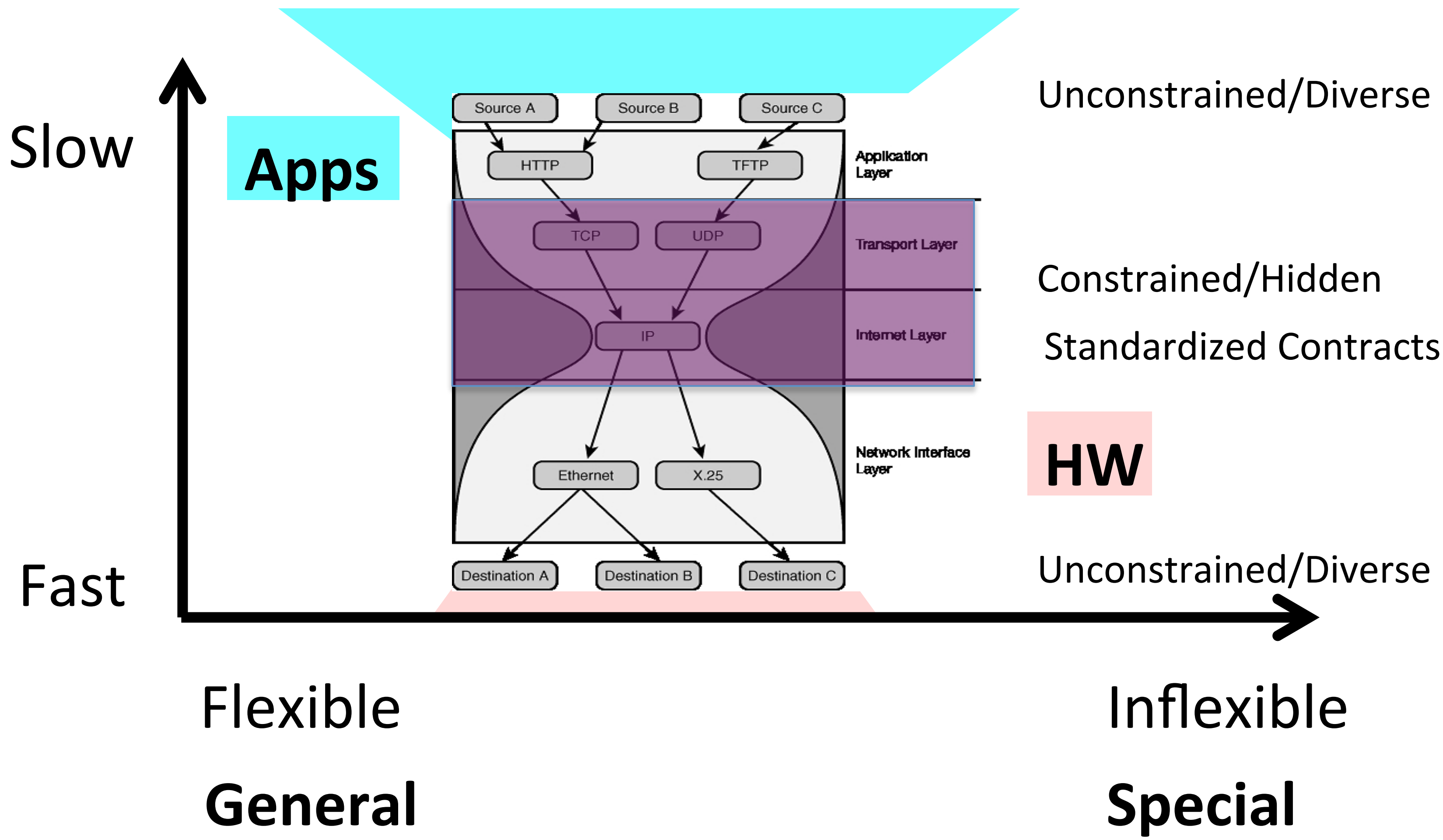
Key Architectural Concept: Horizontal Tran



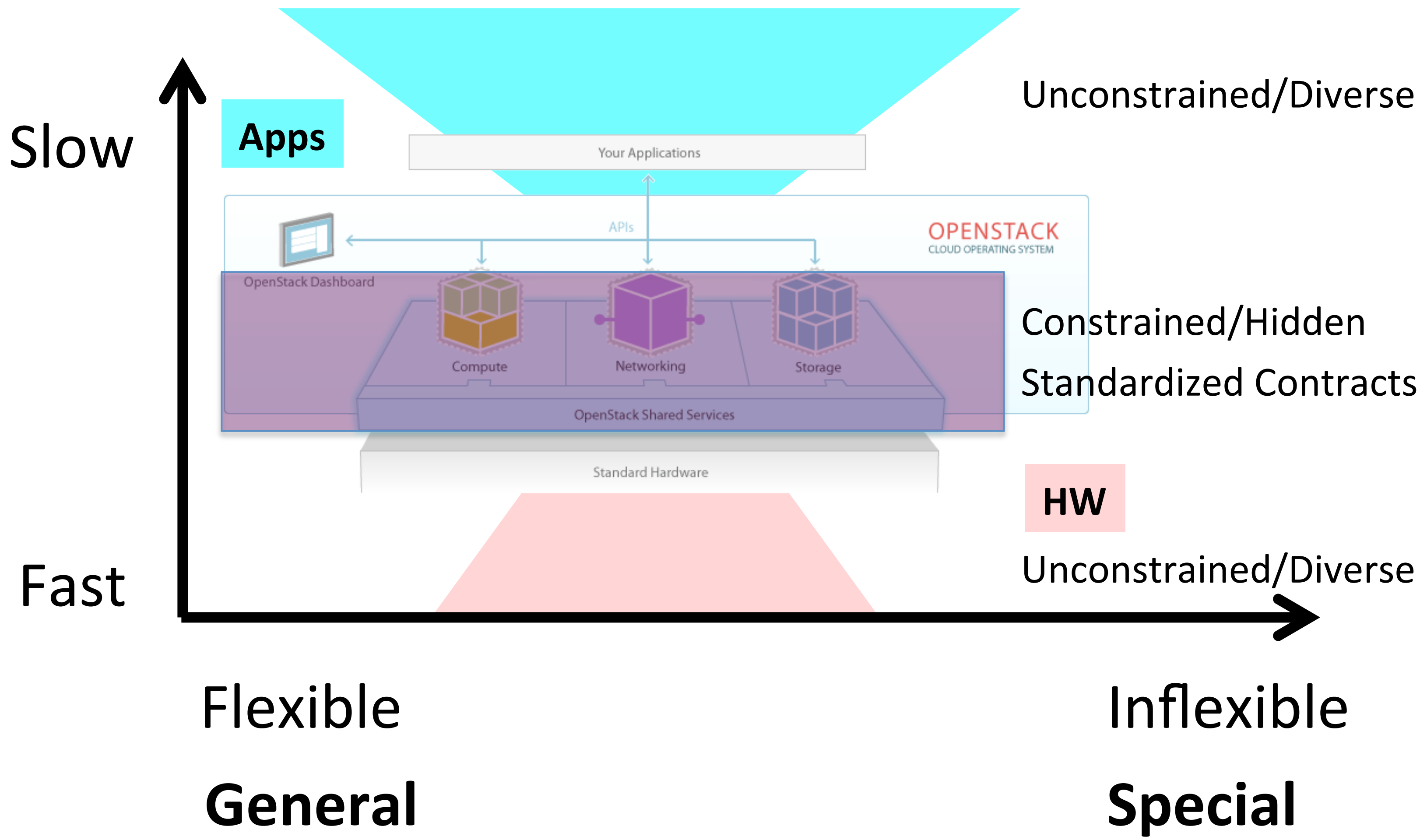
Putting it all Together
Architecture, Layering, and Tradeoffs



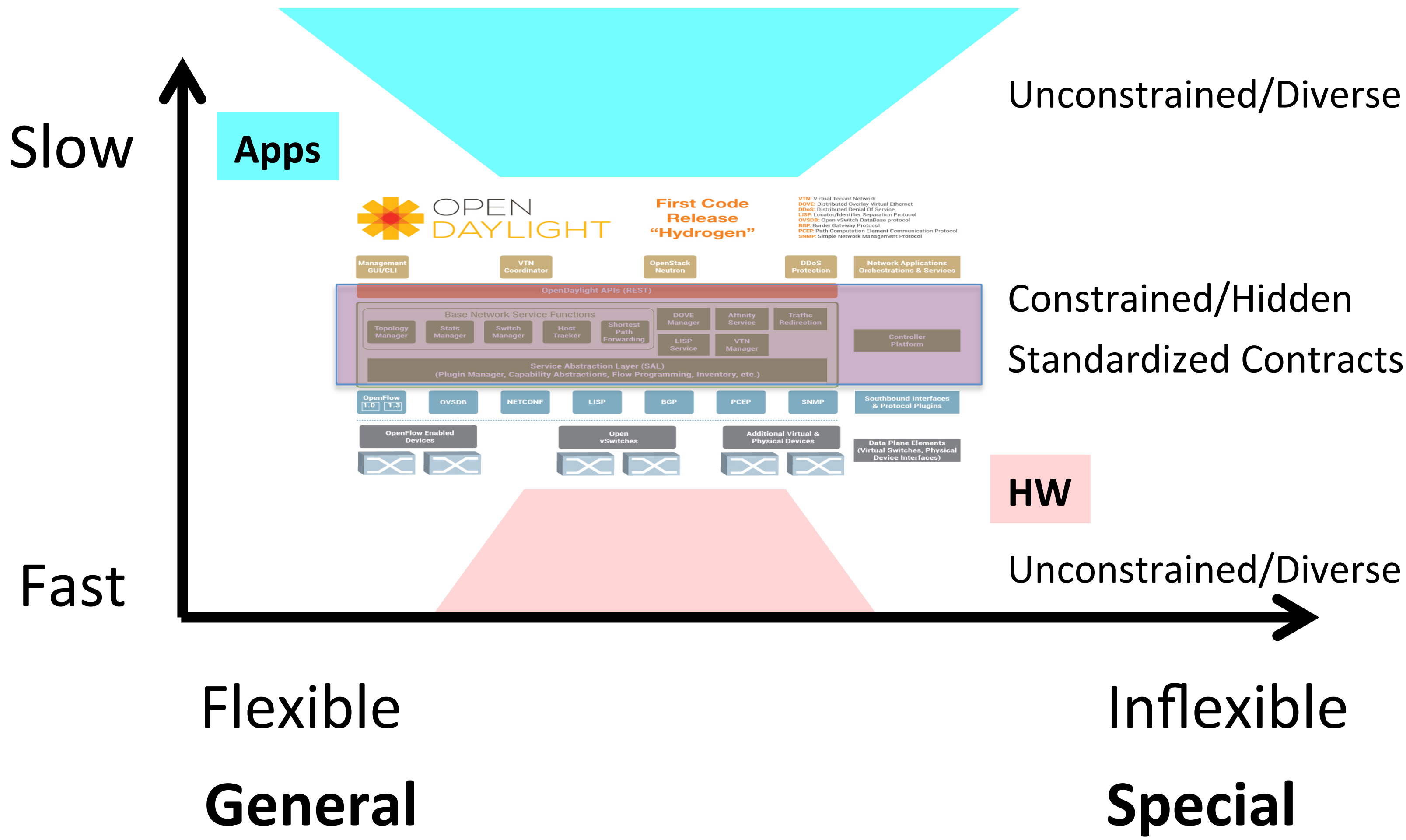
Example: Internet Architecture



Example: OpenStack

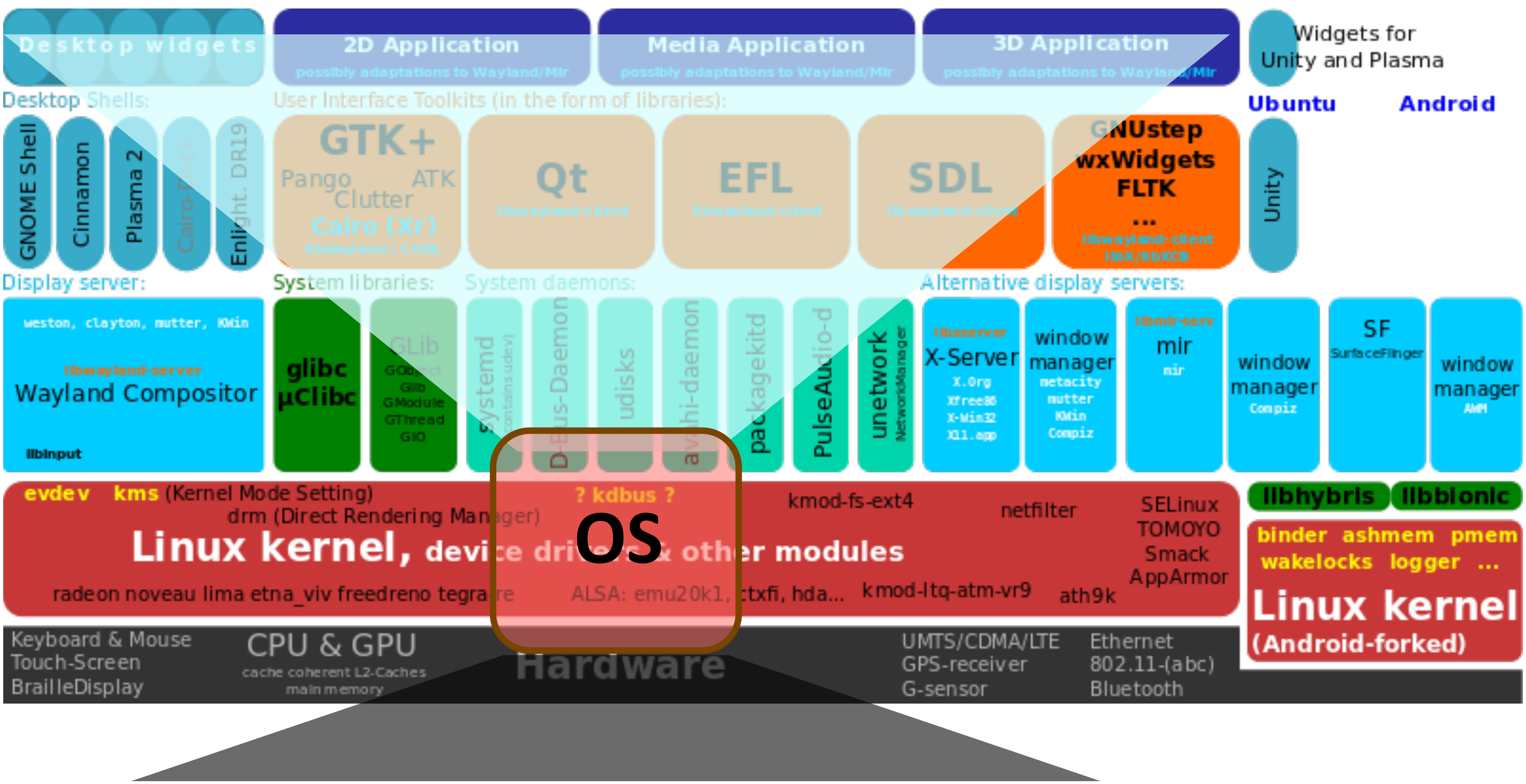


Example: SDN



Slow
Fast
Apps
HW
Unconstrained/Diverse
Constrained/Hidden
Standardized Contracts
Unconstrained/Diverse
Flexible
General
Inflexible
Special
OPEN DAYLIGHT
First Code Release "Hydrogen"
VTN: Virtual Tenant Network
DOVE: Distributed Overlay Virtual Ethernet
DDoS: Distributed Denial Of Service
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch DataBase protocol
BGP: Border Gateway Protocol
PCEP: Path Computation Element Communication Protocol
SNMP: Simple Network Management Protocol
Management GUI/CLI
VTN Coordinator
OpenStack Neutron
DDoS Protection
Network Applications Orchestrations & Services
OpenDaylight APIs (REST)
Base Network Service Functions
Topology Manager
Stats Manager
Switch Manager
Host Tracker
Shortest Path Forwarding
DOVE Manager
Affinity Service
Traffic Redirection
LISP Service
VTN Manager
Controller Platform
Service Abstraction Layer (SAL)
(Plugin Manager, Capability Abstractions, Flow Programming, Inventory, etc.)
OpenFlow 1.0 1.3
OVSDB
NETCONF
LISP
BGP
PCEP
SNMP
Southbound Interfaces & Protocol Plugins
OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices
Data Plane Elements (Virtual Switches, Physical Device Interfaces)

Linux Kernel?

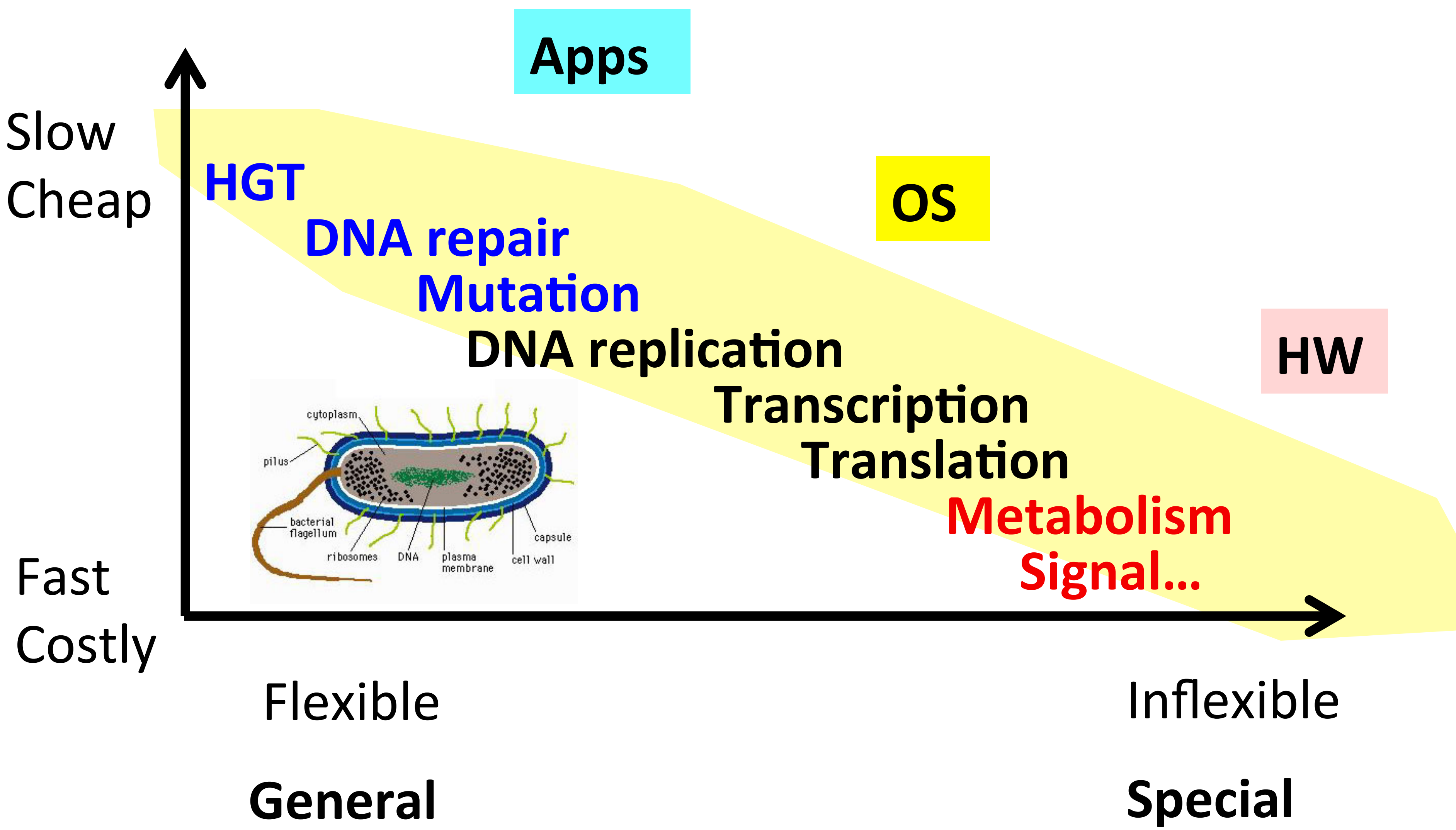


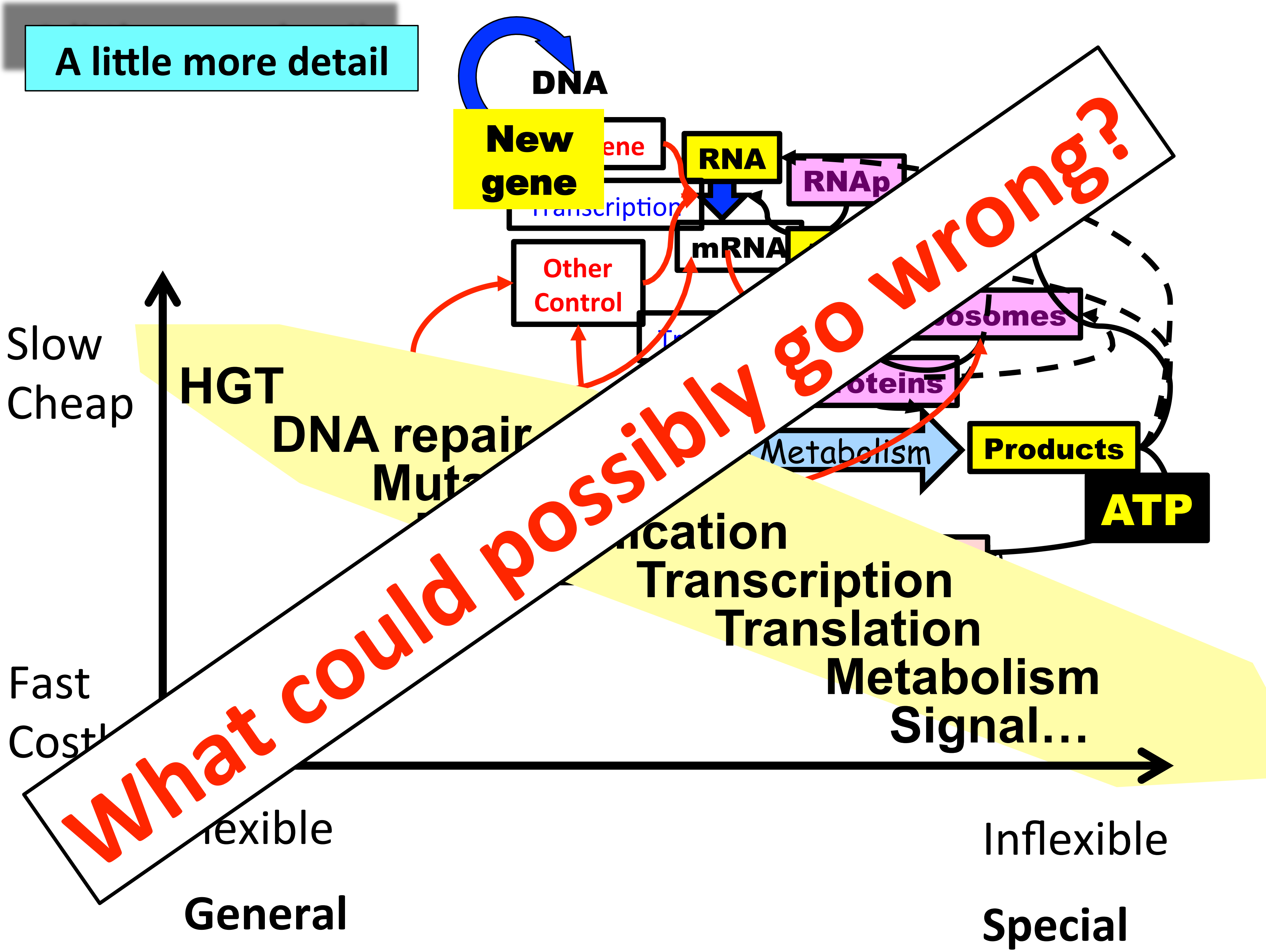
We gotta do some Systems Biology
(come on, its all just networking ☺)

- Biological systems have a similar architecture
  - Same tradeoff space
    - with of course different implementation than our network
  - Basically: diverse apps and h/w with a hidden kernel
  - Constraints that deconstrain

- Many good examples of layered networks
  - Very (brief) examples
    - Bacteria
    - Vestibular Ocular Reflex (VOR)

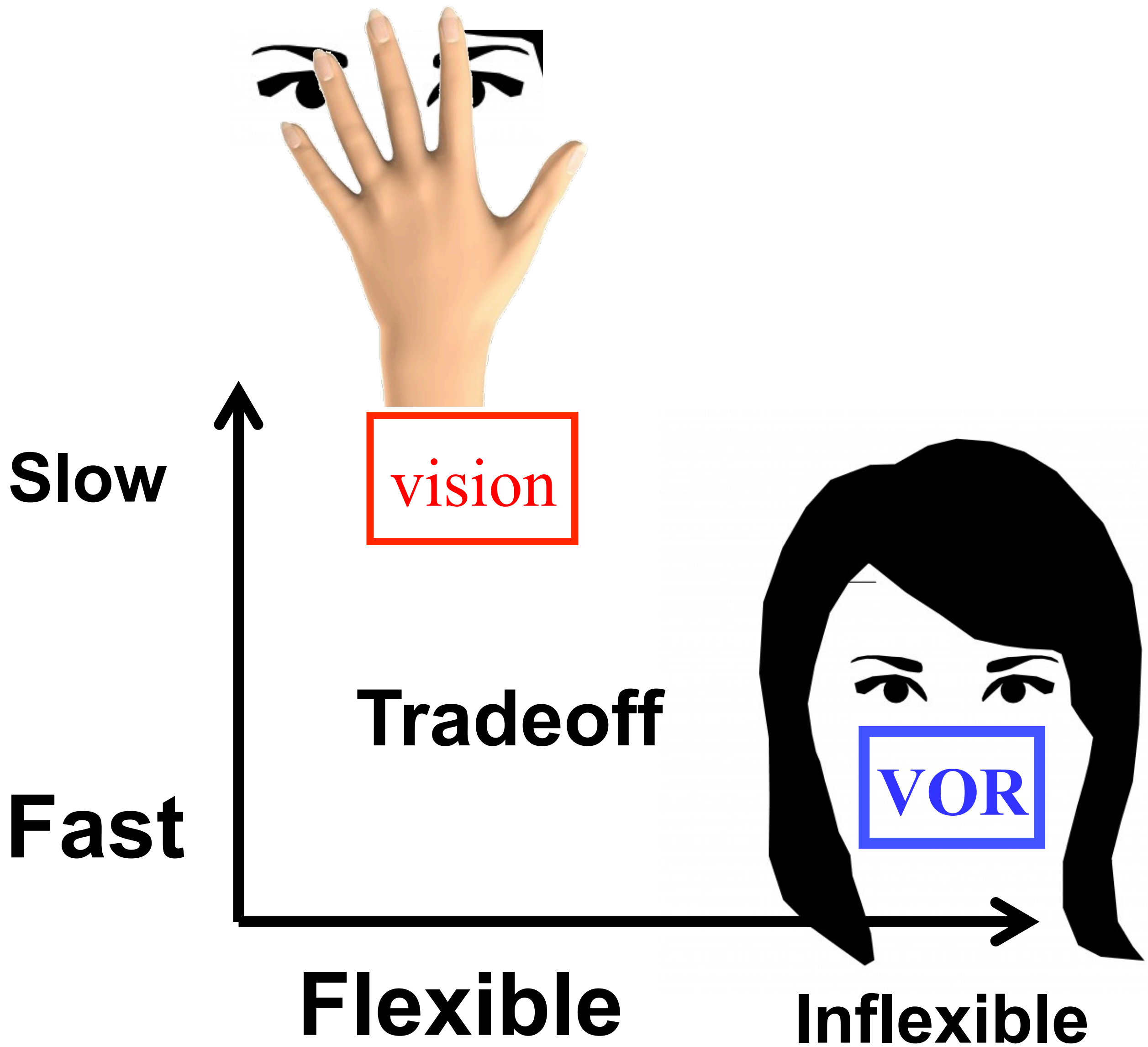
Layered Bacteria




A little more detail
DNA
New gene
ene
RNA
RNAp
mRNA
Other Control
osomes
oteins
Metabolism
Products
ATP
Slow
Cheap
HGT
DNA repair
Muta
ication
Transcription
Translation
Metabolism
Signal…
Fast
Cost
exible
Inflexible
General
Special
What could possibly go wrong?

One More Example: The Vestibular Ocular Reflex (VOR)

Reflex eye movement that stabilizes images on the retina during head movement

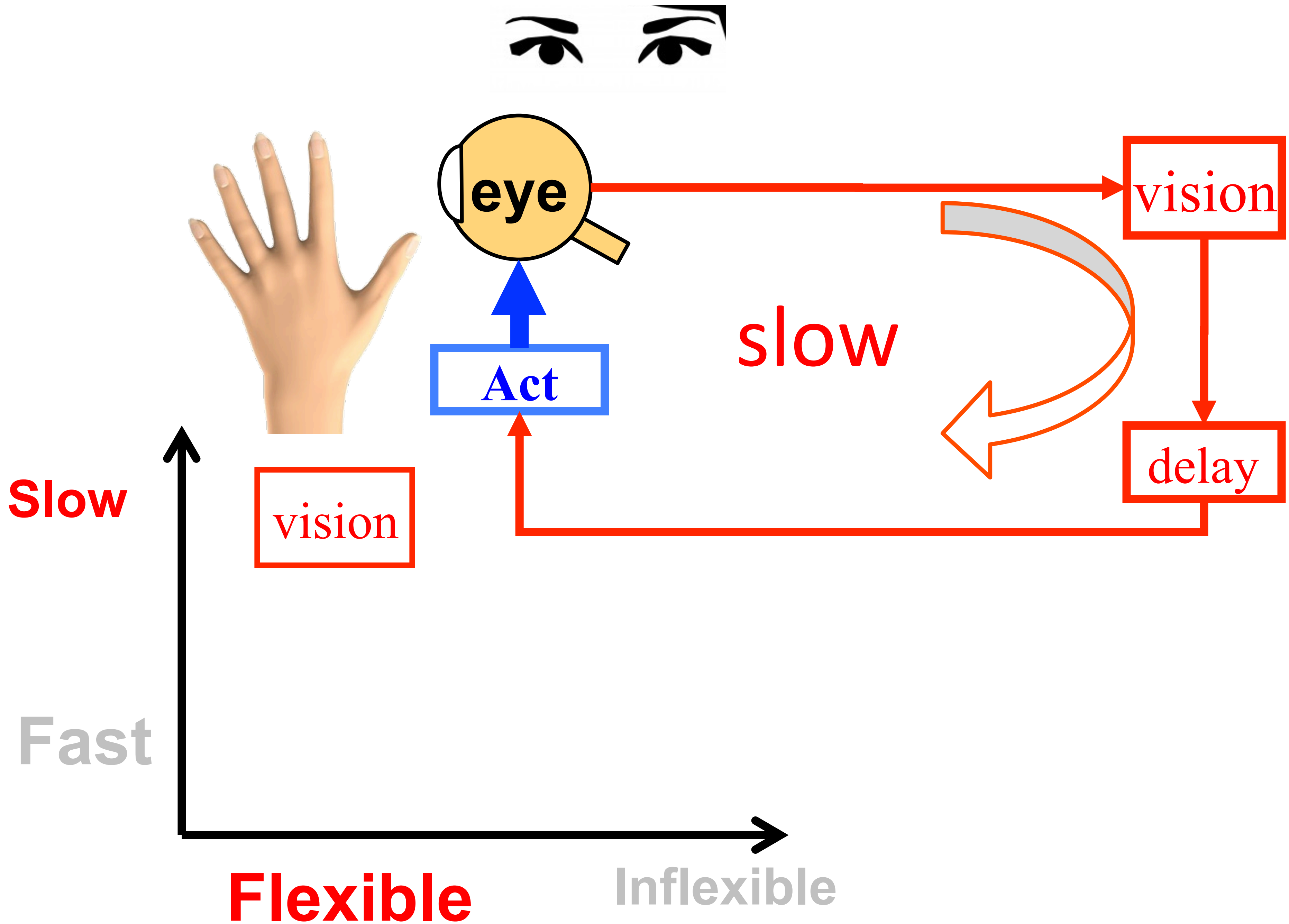


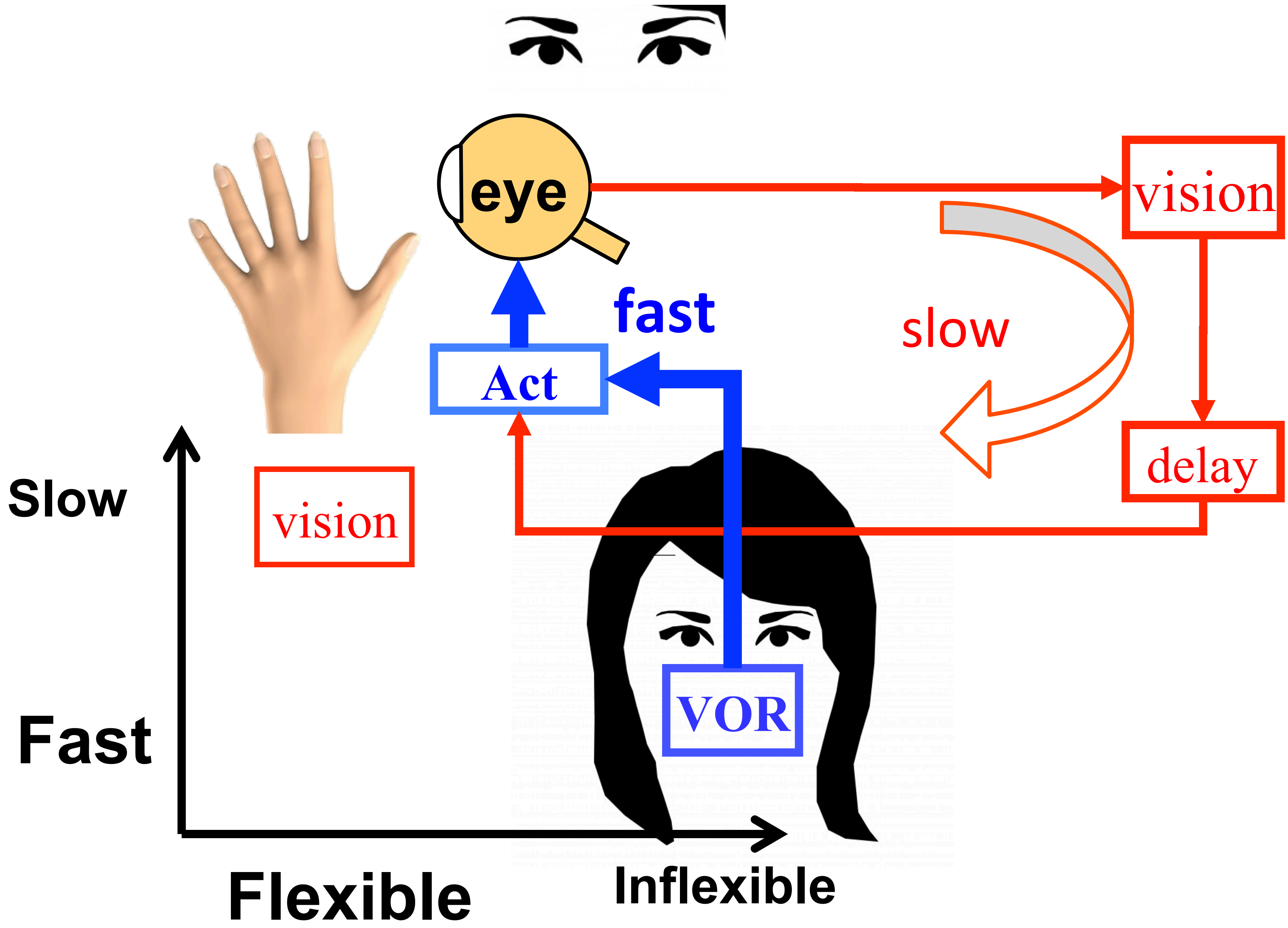
Mechanism

Vestibular

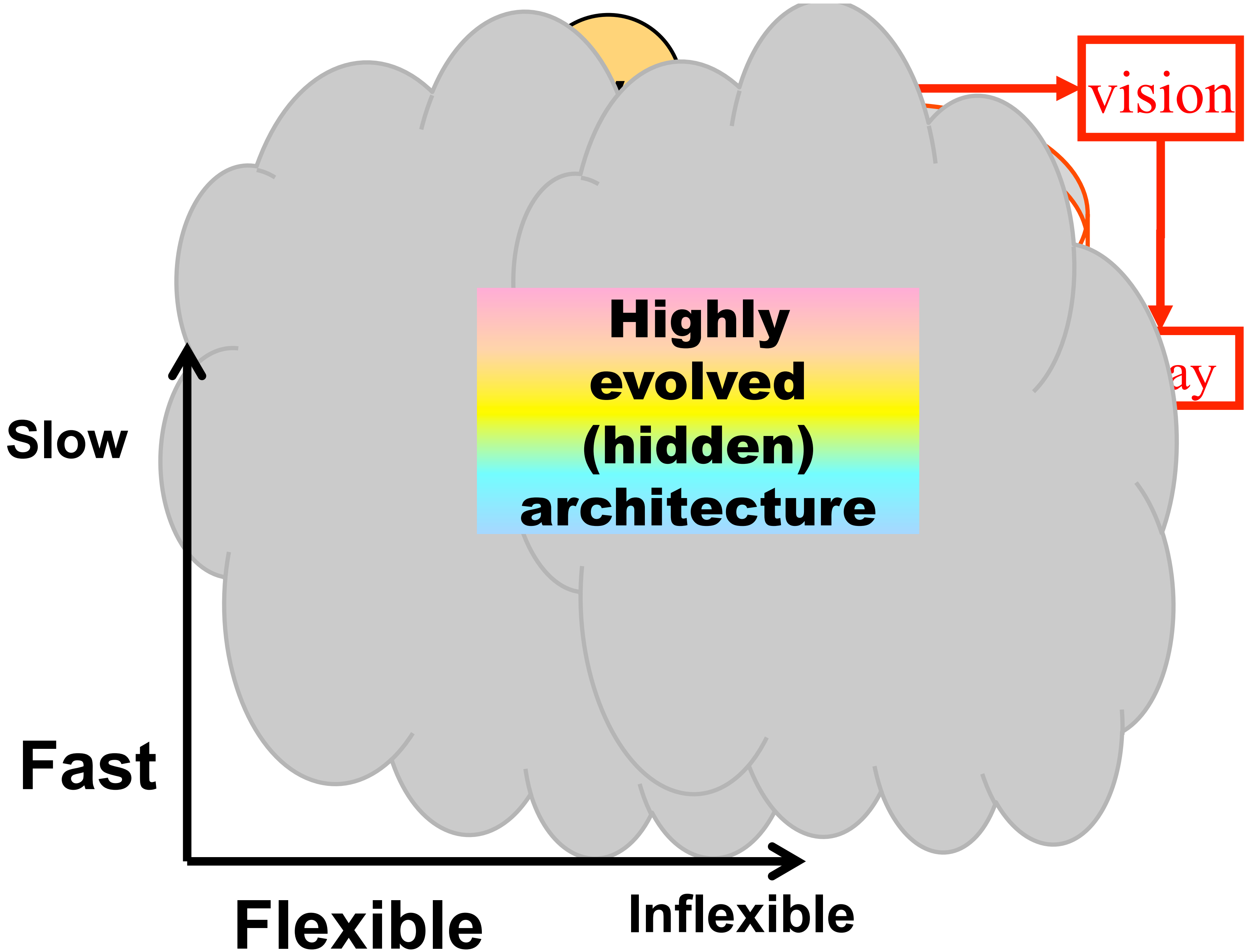
Ocular

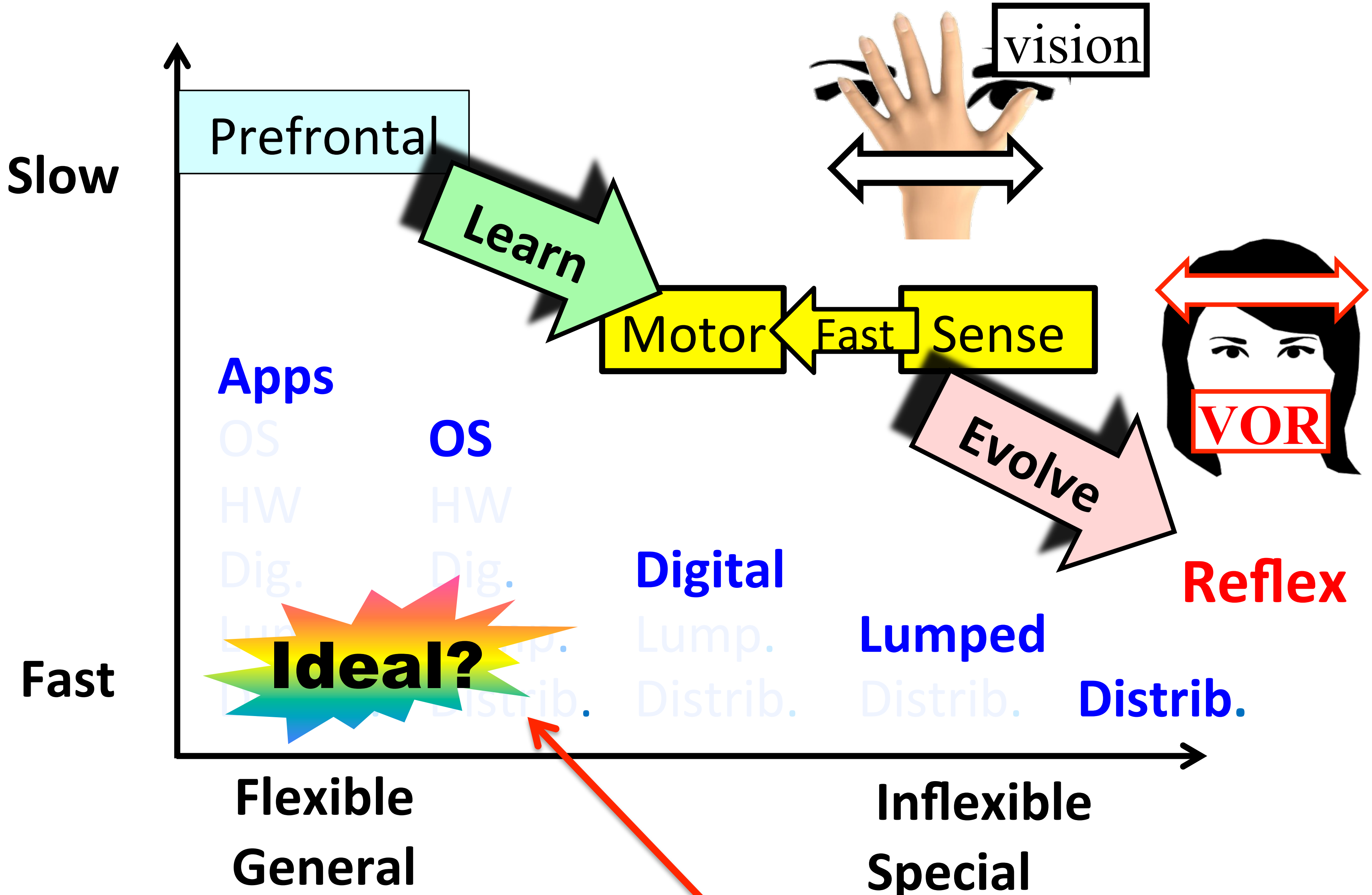
Reflex


eye
vision
slow
Act
delay
Slow
vision
Fast
Flexible
Inflexible

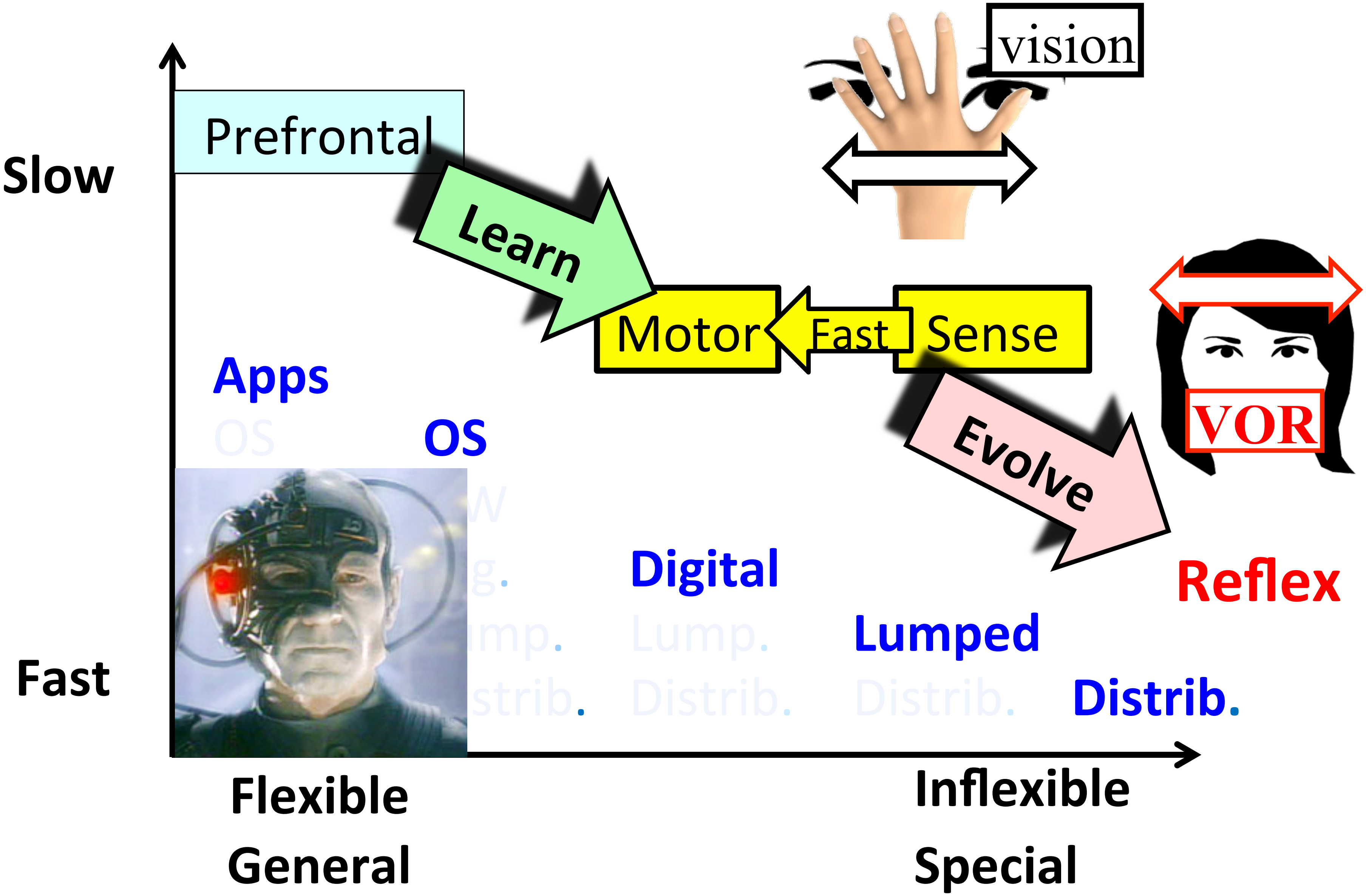


eye
vision
slow
fast
Act
delay
vision
VOR
Slow
Fast
Flexible
Inflexible


vision
ay
Highly evolved (hidden) architecture
Slow
Fast
Flexible
Inflexible



BTW, can a new architecture beat the tradeoff?



Be careful what you wish for…

[*http://packetpushers.net/artificial-intelligence-brains-networks-bugs-complexity/*](http://packetpushers.net/artificial-intelligence-brains-networks-bugs-complexity/)

Agenda

- ~~Too many words, too many slides ☺~~
  - ~~This talk is about thinking about networking in new ways~~

- ~~Motivation and Goals for this Talk~~

- ~~What is Complexity and Why is it Hidden~~
  - ~~Robustness, Fragility, and Complexity~~

- ~~The Architecture of Complex Systems~~
  - ~~Universal Architectural Principles~~

- A Few Conclusions and Q&A if we have time

Hopefully I've Convinced You…

- That there are *Universal Architectural Features* that are common to biology and technology
- Laws, constraints, tradeoffs
  - Robust/fragile
  - Efficient/wasteful
  - Fast/slow
  - Flexible/inflexible
- Architecture/Layering
- Hidden RYF Complexity
- Hijacking, parasitism, predation

- **Ok, but why is this useful?**

Why is all of this Useful?

- Robust systems are intrinsically hard to understand
  - RYF is an inherent property of both advanced technology and biology
    - Understanding general principles informs what we build
  - Software (e.g., SDN, NFV, Cloud, …) exacerbates the situation
  - And the Internet has reached an unprecedented level of complexity →
    - *Need new/analytic ways of designing, deploying, and operating networks if we want to scale*

- Nonetheless, many of our goals for the Internet architecture revolve around how to achieve robustness…
  - which requires a deep understanding of the *necessary interplay between complexity and robustness, modularity, feedback, and fragility*[1]
    - which is neither accidental nor superficial
  - Rather, architecture arises from "designs" to cope with uncertainty in environment and components
  - The same "designs" make some protocols hard to evolve
    - Can anyone say, um, IPv6 (or even DNSSEC)?

[1] See Marie E. Csete and John C. Doyle, "Reverse Engineering of Biological Complexity", http://www.cds.caltech.edu/~doyle/wiki/images/0/05/ScienceOnlinePDF.pdf

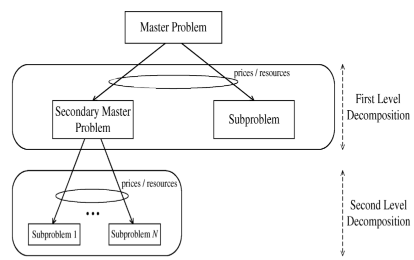
Why is all of this Useful, cont?

- ***This much seems obvious***
  - Understanding these universal architectural features and tradeoffs will help us achieve the scalability and evolvability (operability, deployability, understandability) that we are seeking from the Internet architecture today and going forward

- ***Perhaps less obvious: This requires a mathematical theory of network architecture***
  - Want to be able to analyze/compare/simulate/optimize all aspects of network design/operation
  - Mathematics the natural language for this
  - BTW, as engineers we solve problems ("engineers always know first"), but we can benefit from the tools that theory can provide to help us design/deploy/operate/optimize our networks

- First Cut: "Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures"[1]
  - Network Utility Maximization (NUM)/Layering As Optimization (LAO)/Decomposition Theory

$$
\begin{aligned}
\text{maximize} \quad & \sum_s U_s(x_s, P_{e,s}) + \sum_j V_j(w_j) \\
\text{subject to} \quad & \mathbf{Rx} \le \mathbf{c}(\mathbf{w}, \mathbf{P}_e), \\
& \mathbf{x} \in \mathcal{C}_1(\mathbf{P}_e), \quad \mathbf{x} \in \mathcal{C}_2(\mathbf{F}) \ \text{ or } \in \mathbf{\Pi}(\mathbf{w}), \\
& \mathbf{R} \in \mathcal{R}, \quad \mathbf{F} \in \mathcal{F}, \quad \mathbf{w} \in \mathcal{W}.
\end{aligned}
$$



  - TCP and Stable Path Problem (BGP) reverse-engineered as *Generalized* NUM problems
  - Need something like LAO/G-NUM + "Constraints that deconstrain" view

- Finally, some bridge building: *Exploring the Intersection of Theory and Engineering: Universal Laws, Architecture, and SDN:* http://conferences.sigcomm.org/sigcomm/2014/tutorial-theory+eng.php

1 http://www.princeton.edu/~chiangm/layering.pdf

Q&A

Thanks!