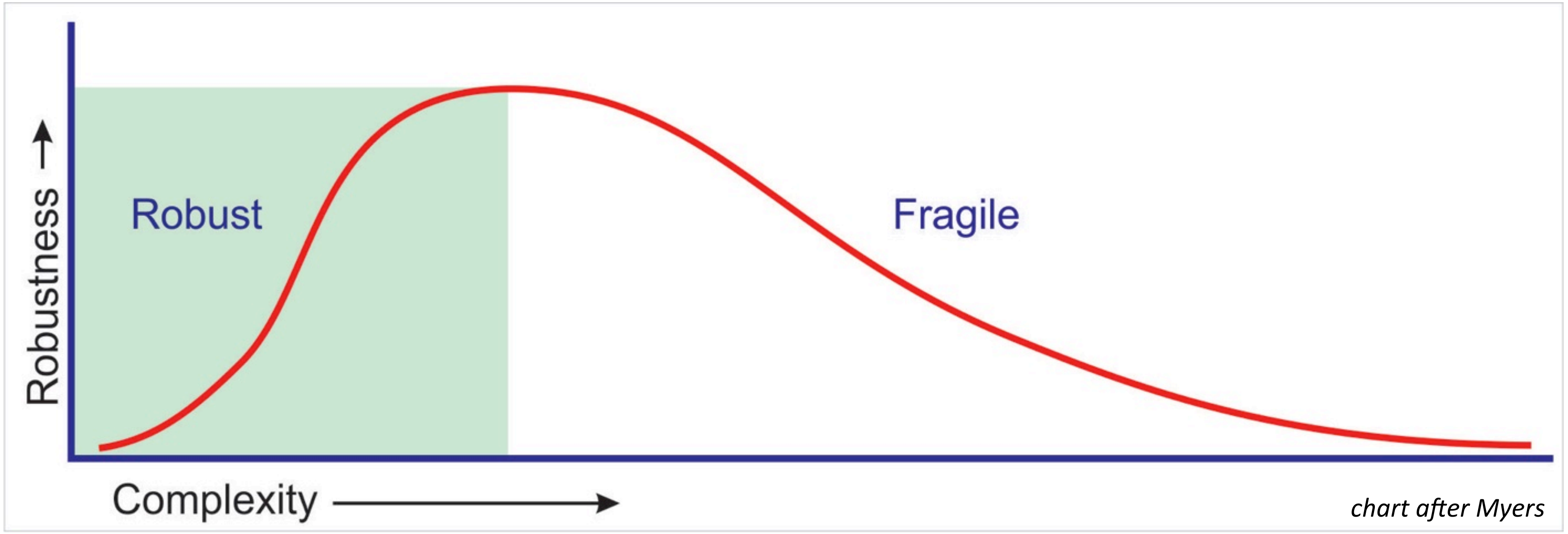
Tradeoffs in Network Complexity

Introducing Complexity

- “It won’t scale”
- “It’s not elegant”
- These are statements about complexity
- “Reduce complexity,” right?

Network Complexity is Necessary

- Complexity is necessary to build robust systems
- We can't get away from complexity…



Network Complexity is Necessary

“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, “ContrasDng Views of Complexity and Their ImplicaDons For Network---Centric Infrastructures”, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 40, NO. 4, JULY 2010

Complexity is Impossible to Solve

- We cannot get to the lower left hand corner
- There is a “sweet spot,” where the optimal robustness is paired with the lowest cost
- RYF, CAP, QSC, etc. are all symptoms of this problem

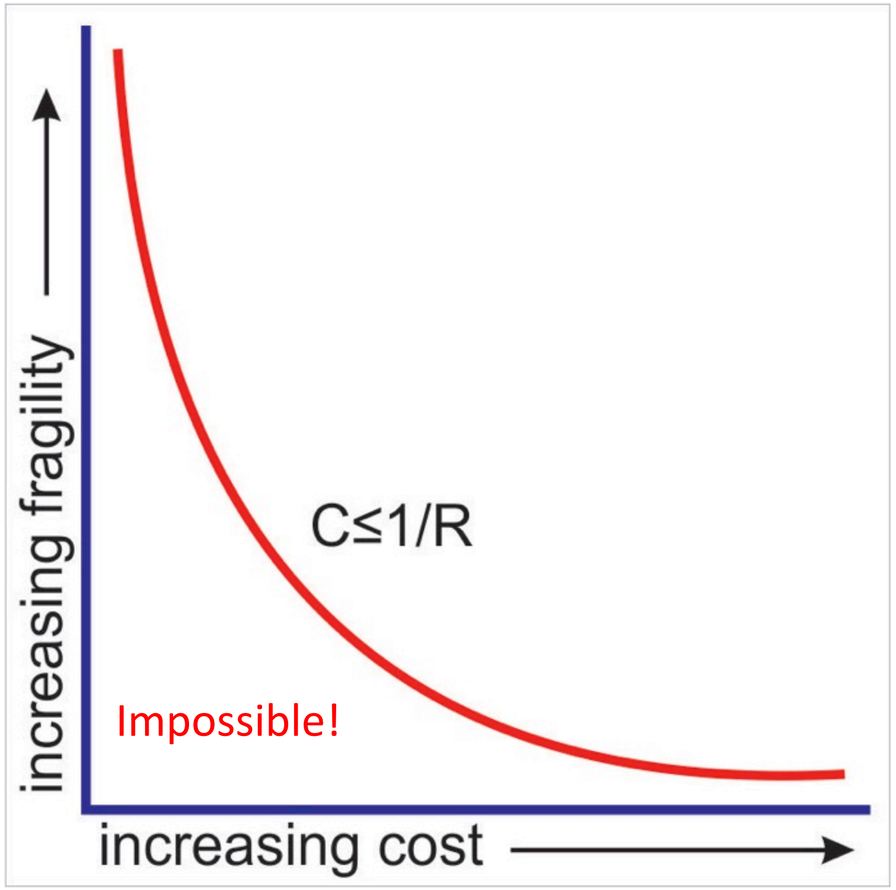
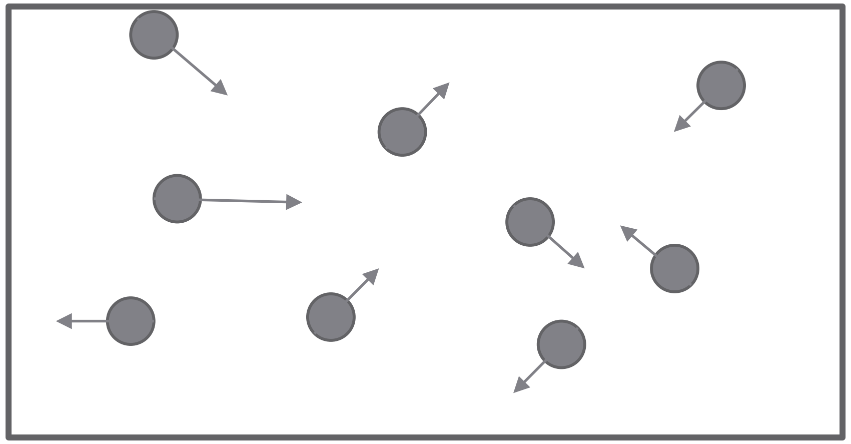
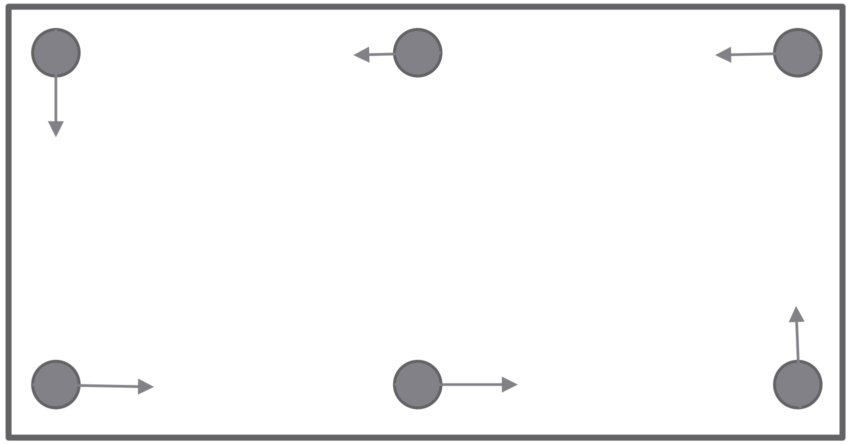


chart after Myers

Network Complexity is Organized

- Organized complexity is different than disorganized complexity
- *They are all problems which involve dealing simultaneously with a sizable number of factors which are interrelated into an organic whole. They are all, in the language here proposed, problems of organized complexity. –Weaver, 1948*





Network Complexity is Organized

- Statistical models will be of limited use in this realm
  - Statistics will tell you if there is information (Shannon), but not what that information means
- We must interact with *intent*
  - Network design *intends* to solve specific problems
  - How can you measure intent?

So…

- Complexity in networks is necessary
- Complexity in networks is impossible to solve (perfectly)
- Complexity in networks is difficult (or impossible) to measure and characterize
- Abandon hope all ye who enter here?

- *Or maybe…*

Complexity as a Tradeoff

Complexity as a Tradeoff

- Maybe we look at complexity as a set of tradeoffs
  - Increasing complexity to increase robustness at point a will lead to decreased robustness at point b
  - Reducing complexity at point a will cause complexity at point b to rise
- So…
  - Find the tradeoffs and describe them
  - Find ways to measure both sides of each of the tradeoff
  - Make better (more intelligent) decisions about design
- Let’s consider a few examples

Complexity verses the Problem

- Harder problems tend to require more complex solutions
    - Complexity has no meaning outside the context of the problem being solved
    - Nail verses screw verses screw+glue
- How many balloons fit in a bag?

Complexity verses the Toolset

- More complexity can be managed with better tools
  - If your only tool is a hammer…
- But we need to figure in the cost of the tool
  - Nail guns are harder to maintain than hammers
  - Sonic screwdrivers are notorious for breaking at just the wrong moment

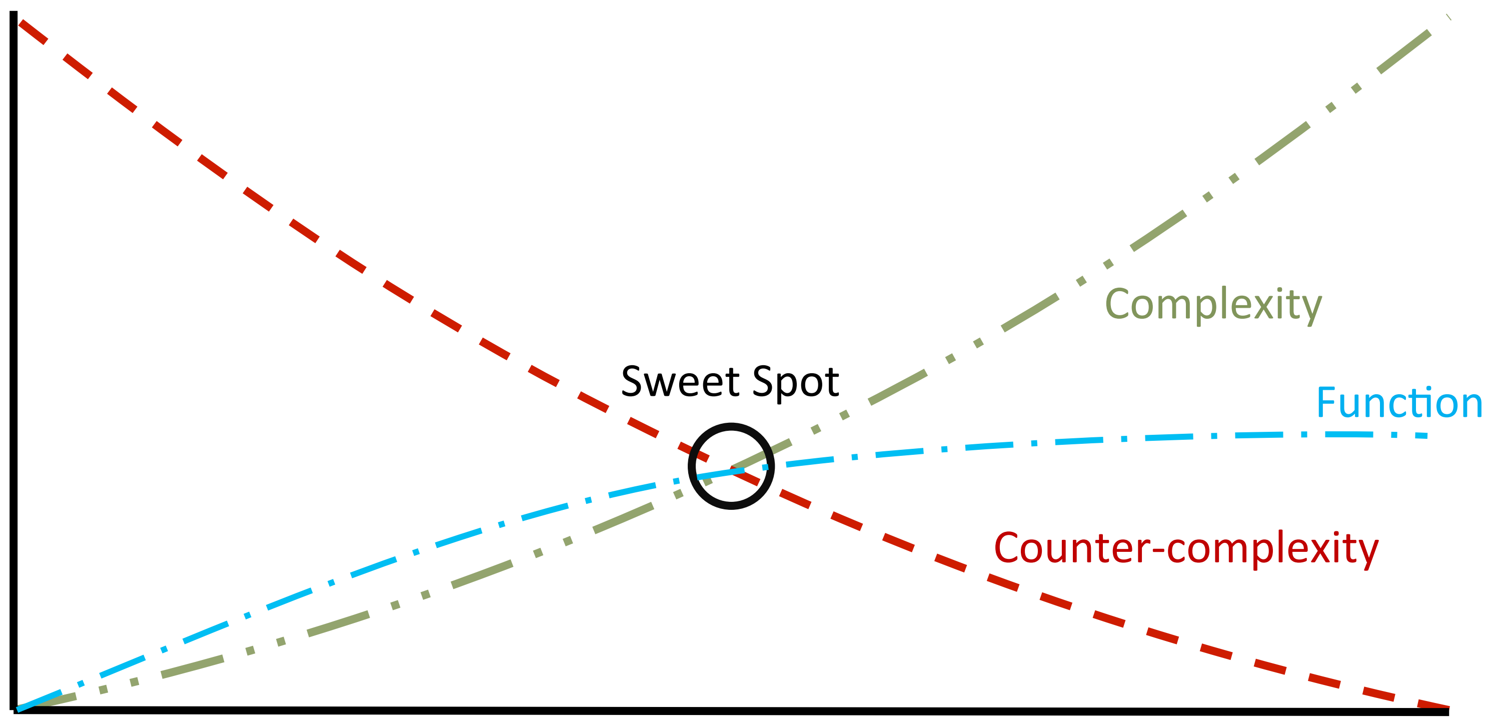
Complexity verses Skill Set

- Things that are complex for one person might not be for another…
  - This isn't a (just) matter of intelligence, it's also a matter of focus and training

Complexity verses Complexity

- Complexity comes in pairs
  - *It is easier to move a problem around (for example, by moving the problem to a different part of the overall network architecture) than it is to solve it.*
  - *It is always possible to add another level of indirection.*
  - RFC1925
- Decreasing complexity in one part of the system will (almost always) increase complexity in another

The Complexity Graph



Simple!

The Point

- You can never reach some other desirable goal without increasing complexity
  - Decreasing complexity in one place will (nearly) always increase it in another
  - Decreasing complexity in one place will often lead to suboptimal behavior in another
  - Increasing service levels or solving hard problems will almost always increase complexity

You don't have to have a point, to have a point…

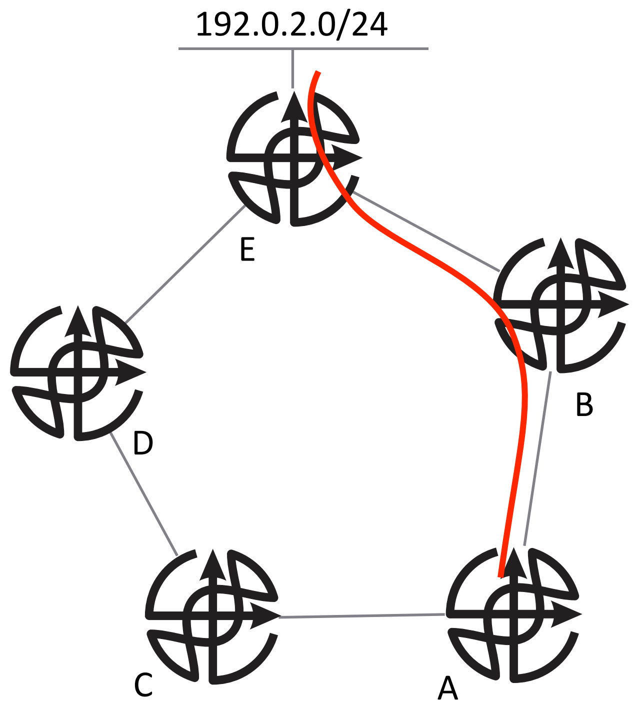
The Goal

- Bad questions
  - How complex is this?
  - Will this scale?
- Good questions
  - Where will adding this new thing increase complexity?
  - If I reduce complexity here, where will I increase it?
  - If I reduce complexity here, where will suboptimal behavior show up?
- Complexity at the system level is about *tradeoffs,* not *absolutes*

Fast Reroute as an Example

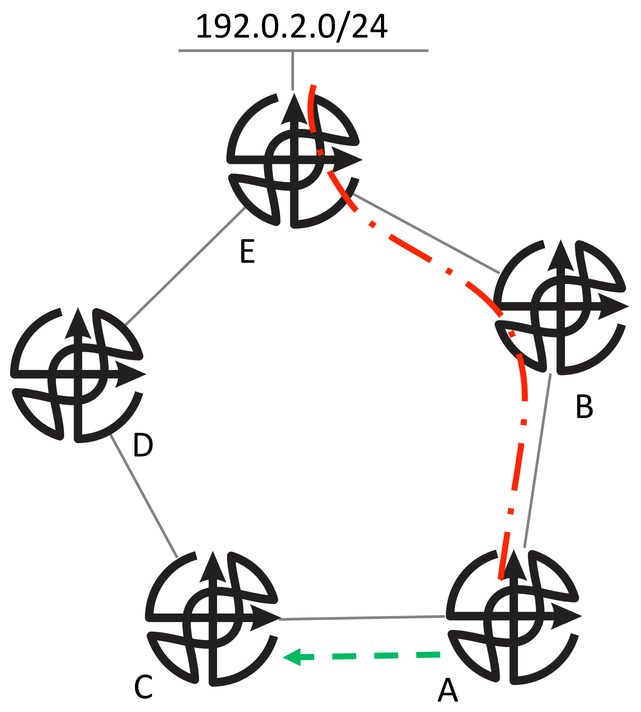
Precompute

- Router A uses the path through B as its primary path to 192.0.2.0/24
- There is a path through C, but this path is blocked by the control plane
  - If A forwards traffic towards 192.0.2.0/24 to C, there is at least some chance that traffic will be reflected back to A, forming a routing loop
- We would like to be able to use C as an alternate path in the case of a link failure along A->B->E



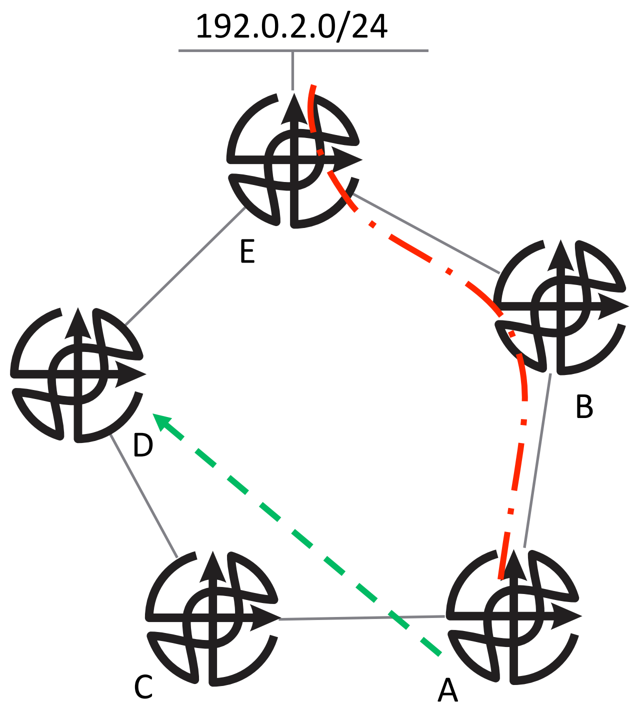
Precompute: LFAs

- Loop Free Alternates (LFAs)
  - A can compute the cost from C to determine if traffic forwarded to 192.0.2.0/24 will, in fact, be looped back to A
  - If not, then A can install the path through C as a backup path
- Gains
  - Faster convergence
- Costs
  - Additional computation at A *(almost nil)*
  - Designing the network with LFAs in mind



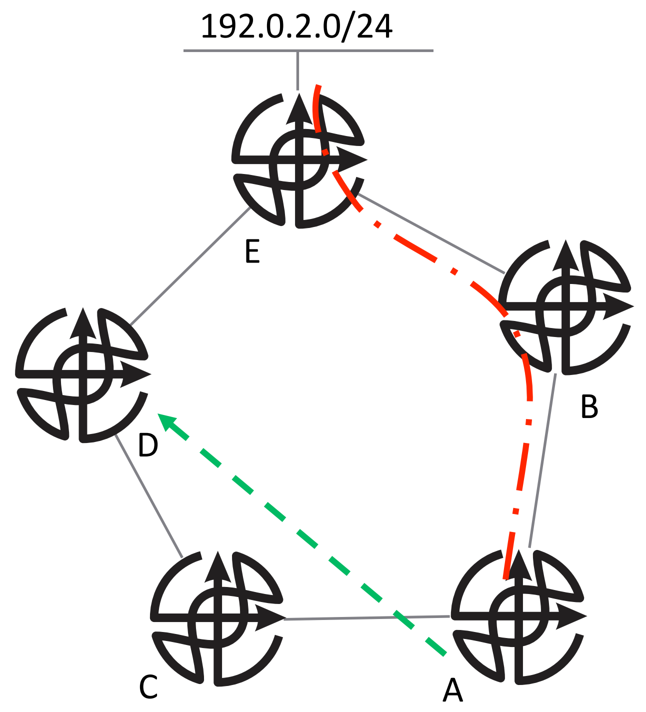
Precompute: Tunneled LFAs

- Tunnel into Q
  - A can compute the first hop beyond C where traffic destined to 192.0.2.0/24 will not loop back
  - A then dynamically builds a tunnel through C to this point and installs the tunnel interface as a backup route
  - There are a number of ways to do this
    - NotVIA, MRT, Remote LFA, etc.
    - Different computation and tunneling mechanisms, but the general theory of operation is the same



Precompute: Tunneled LFAs

- Gains
  - Relaxed network design rules (rings are okay)
  - Eliminates microloops
  - Faster convergence
- Costs
  - Additional computation at A *(almost nil)*
  - Some form of dynamic tunnel
  - Additional control plane state
  - Designing the network with alternate paths in mind
    - These mechanisms don't support every possible topology (but more than LFAs)
    - Thinking about alternate traffic patterns to project link overload, QoS requirements, etc.



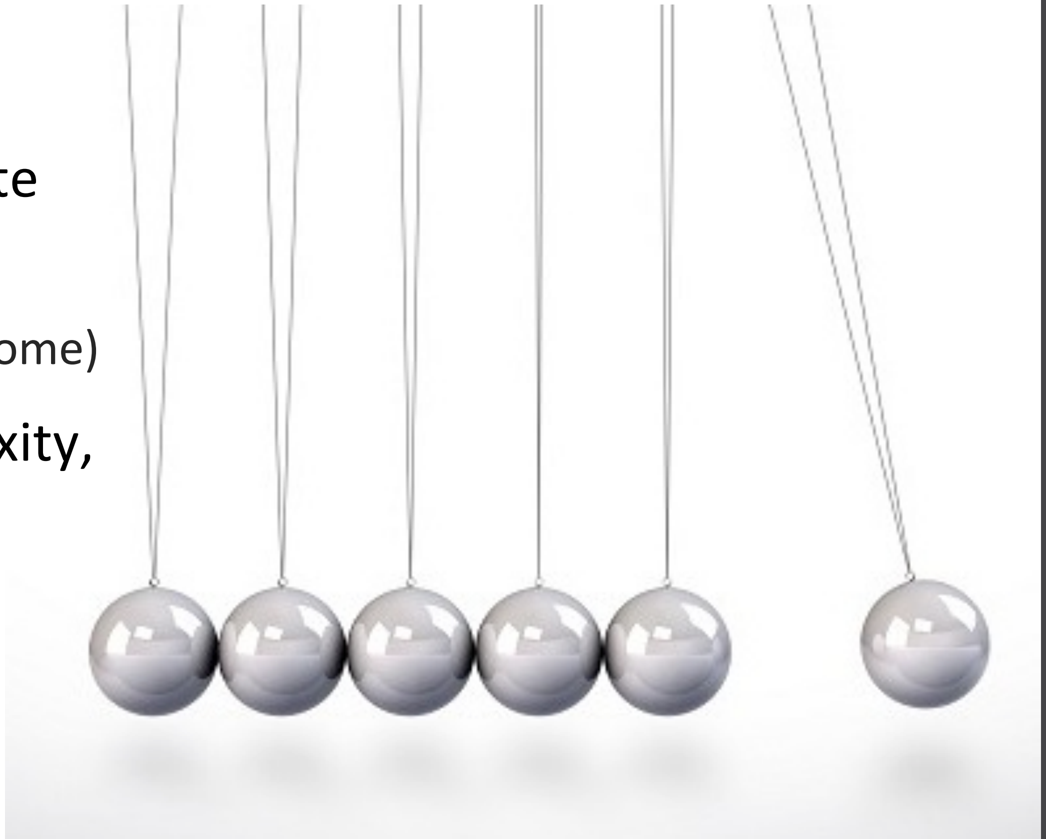
Whither Complexity?

Whither Complexity?

- Will we ever have a single number that tells us how complex a network is?
  - Probably not
  - But we *will* have a bunch of numbers that help us characterize specific parts
- Will we ever have something we can point to that will mathematically prove, “this is complex,” “that won’t scale,” etc.?
  - No…
  - But we can understand what complexity looks like so we can “see” elegance more clearly

Whither Complexity?

- One useful result would be a more realistic view of network design and operation
- We're caught on multiple pendulums
  - Centralize! Decentralize!
  - Layer protocols! Reduce protocol count!
- Most of these swings relate to our absolute view of complexity
  - There *must* be a better solution!
  - Let's go try that over there! (shiny thing syndrome)
- If we could gain a realistic view of complexity, we might be able to see how to at least reduce the frequency and amplitude…



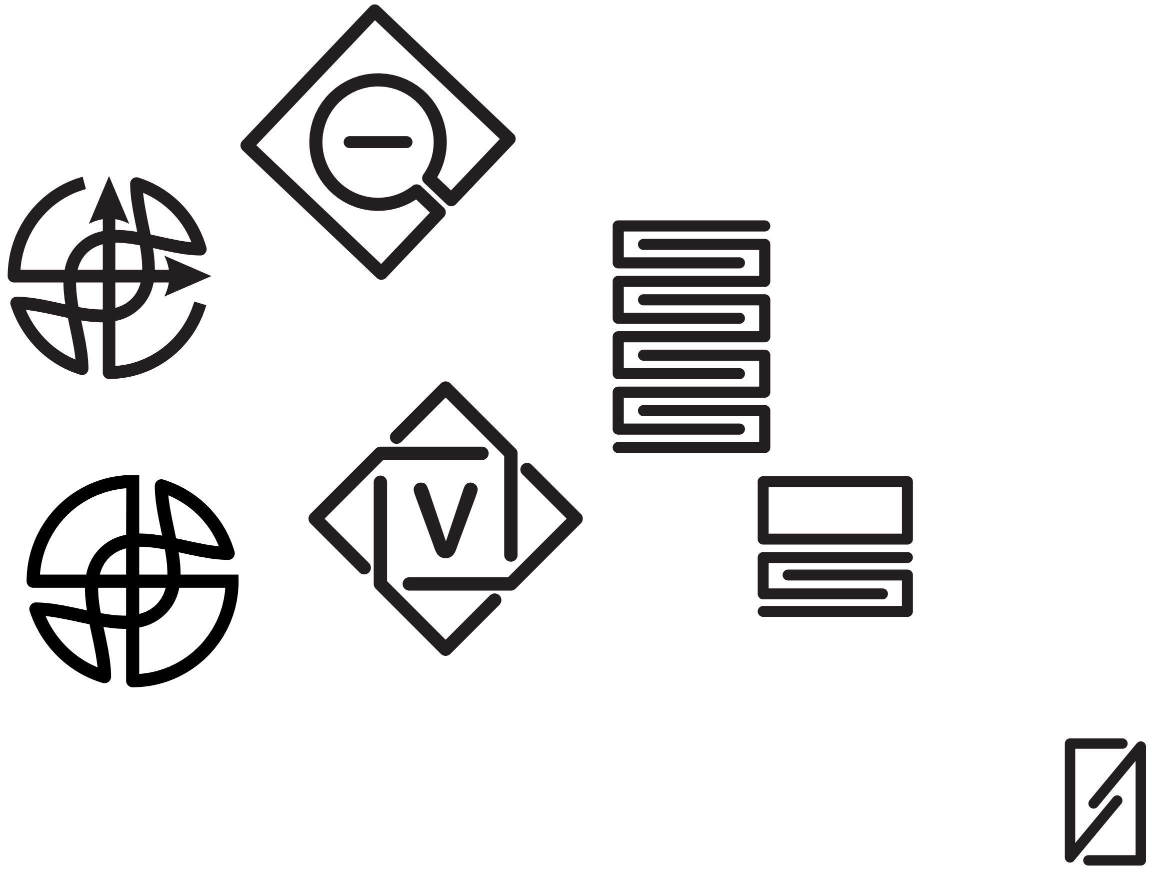
One Way Forward

- Measurements within a framework
  - Understand the system as a whole
  - Think about how to measure each point
  - Think about how to compare, or weigh, each pair of points
- Document the tradeoffs we find in real life
  - Helps guide the work of developing measurements
  - Helps build a “body of knowledge” that will drive the state of the art in network design forward

Efforts to Measure & Describe

- Network Complexity Working Group (NCRG)
  - IRTF working group
  - Trying to find ways to describe and measure complexity
  - Gathering papers in the network complexity space on networkcomplexity.org
- draft-irtf-ncrg-network-design-complexity-00.txt
  - Within NCRG
  - Parallel to this presentation
- Many other efforts afoot

The End!


V