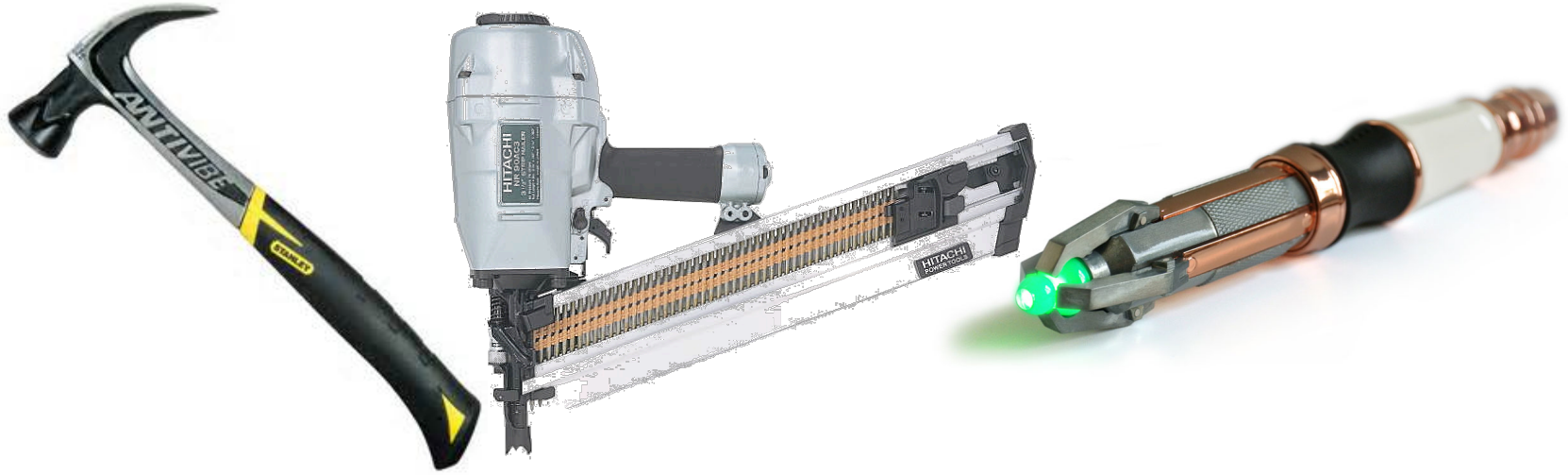
Tradeoffs in Network Complexity

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



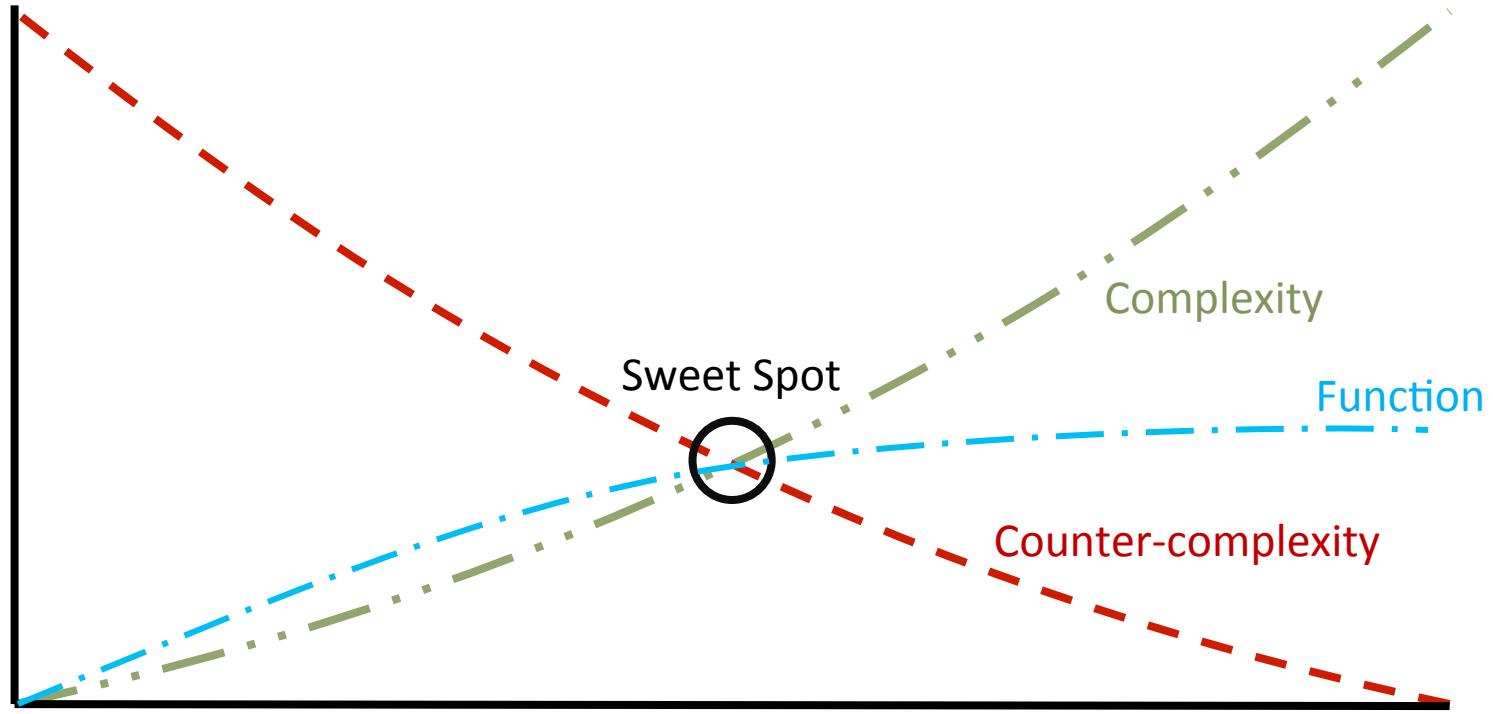
Complexity verses Skill Set

- 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…

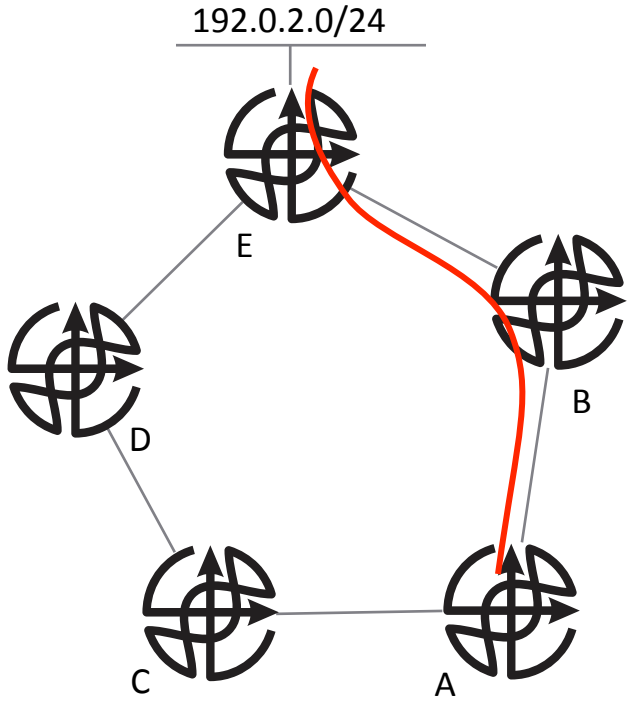
The Goal

- 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

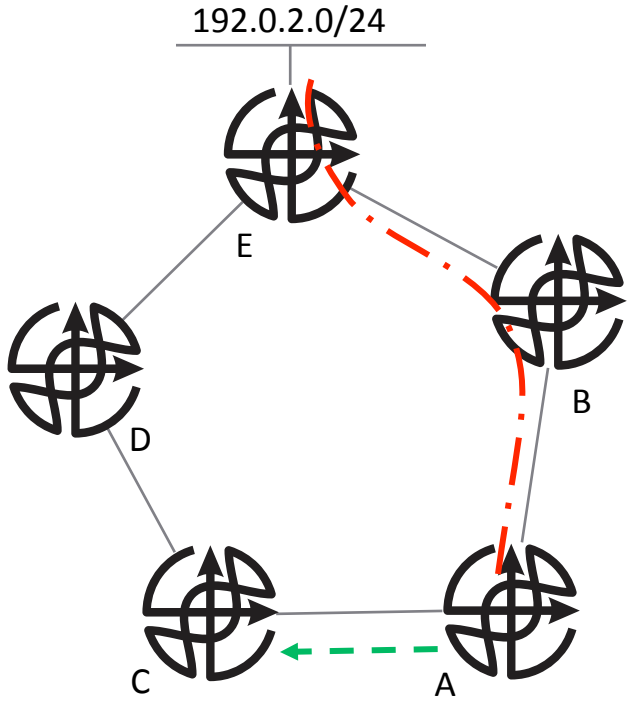
Precompute

- 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



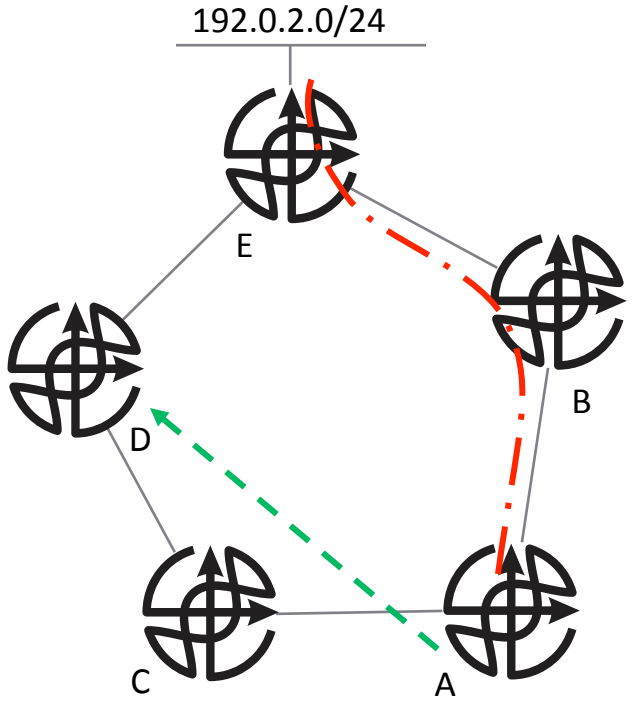
Precompute: LFAs

- 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



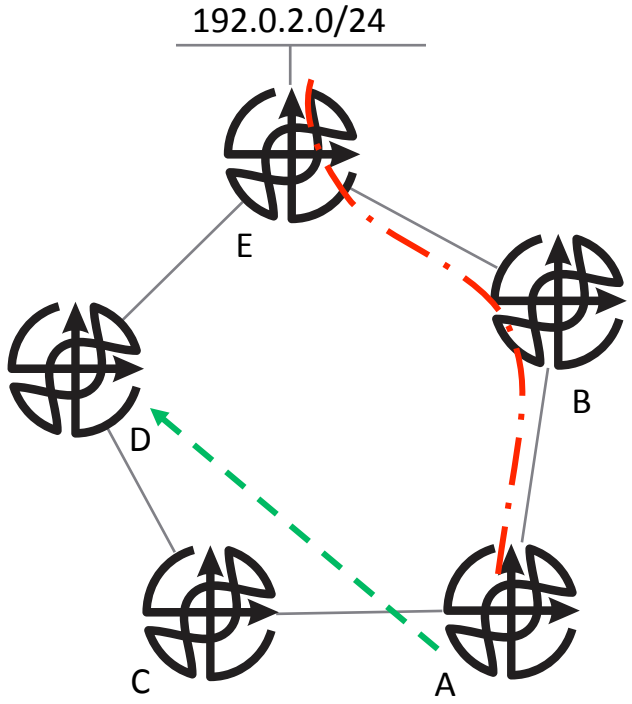
Precompute: Tunneled LFAs

- 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.



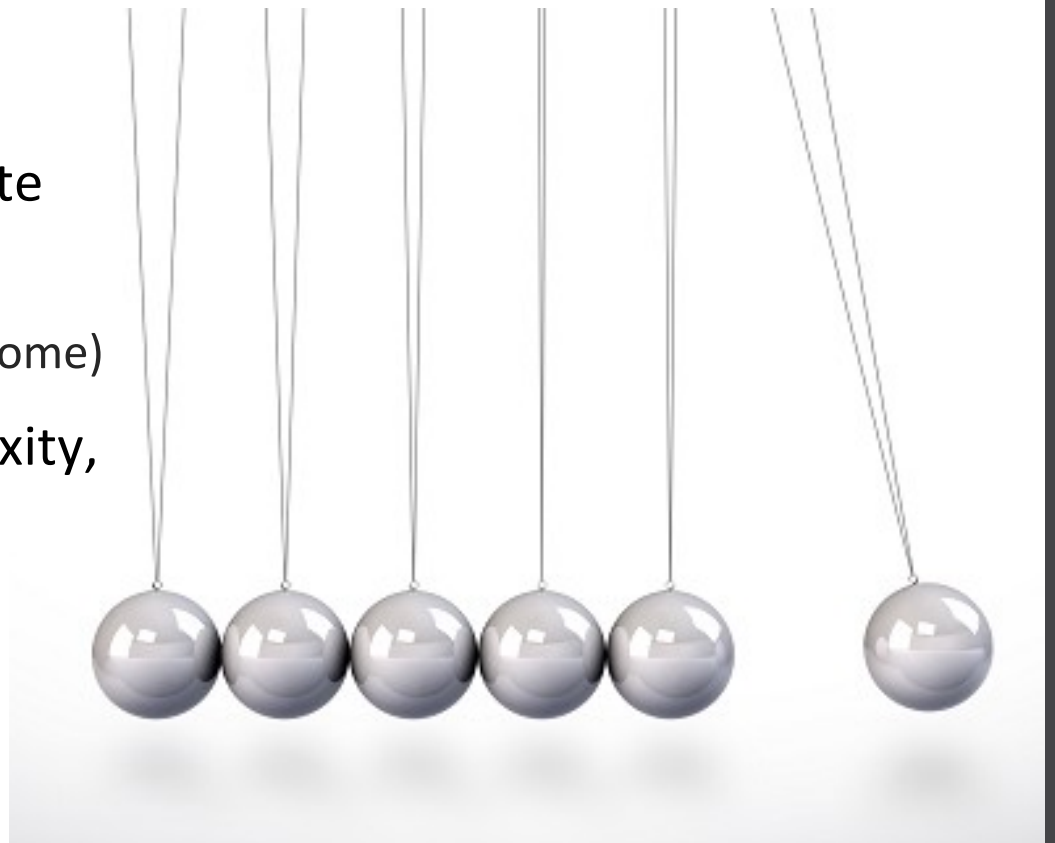
Whither Complexity?

Whither Complexity?

- Will we ever have a single number that tells us how complex a network is?
  - No…
  - 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…



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
  - This is so complex – 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 of these pendulum swings…

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

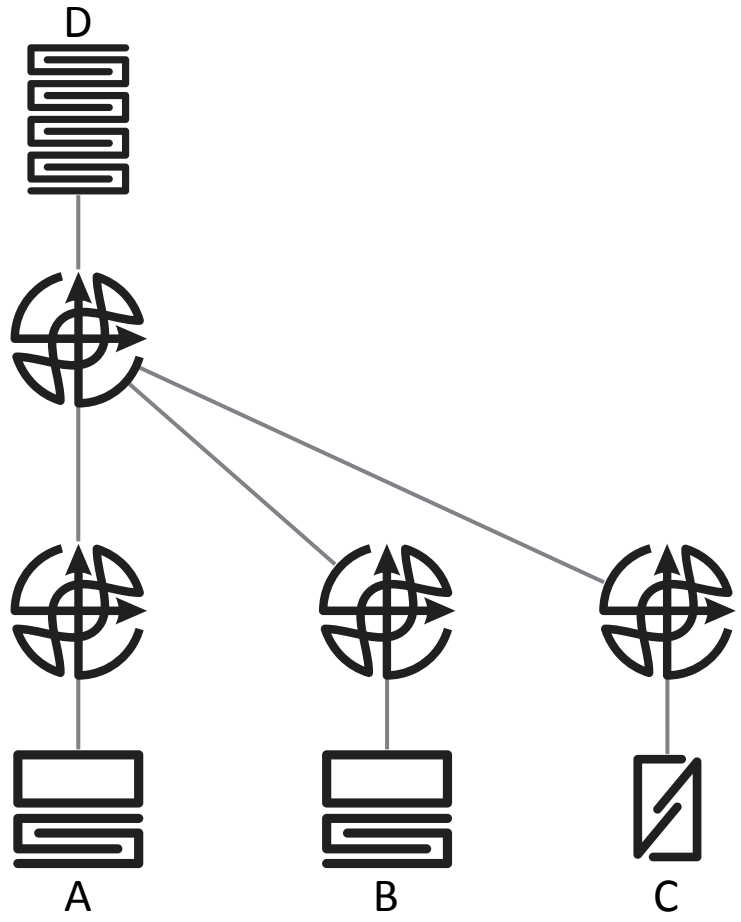
Efforts to Measure & Describe

- 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

Policy Dispersion Example

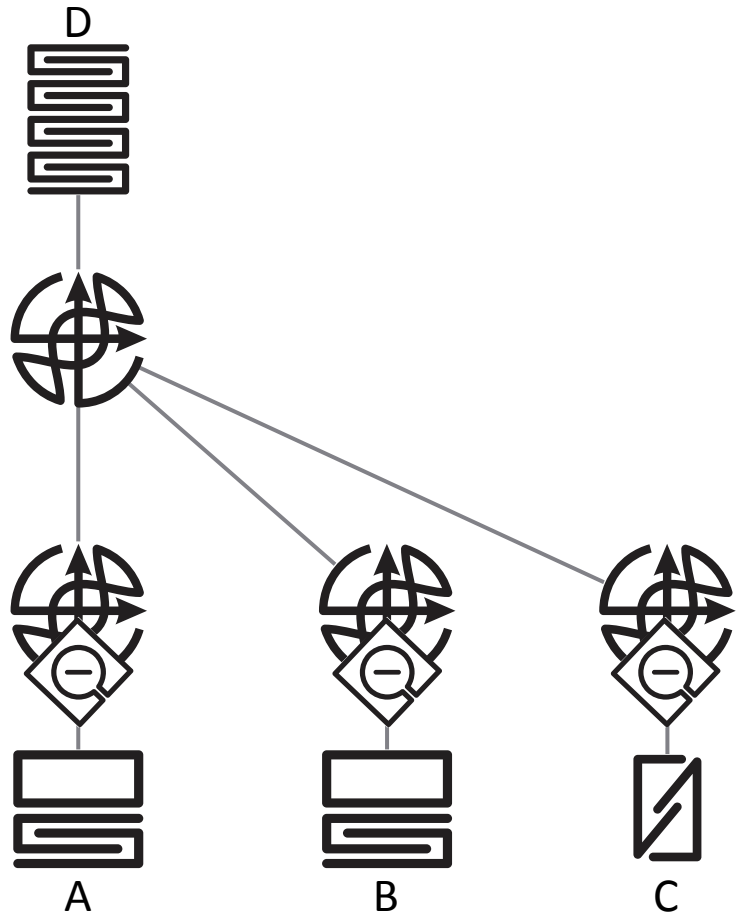
Optimal Forwarding

- Traffic originating at A, B, and C must pass through deep packet inspection before reaching D
- Where should we put this policy?



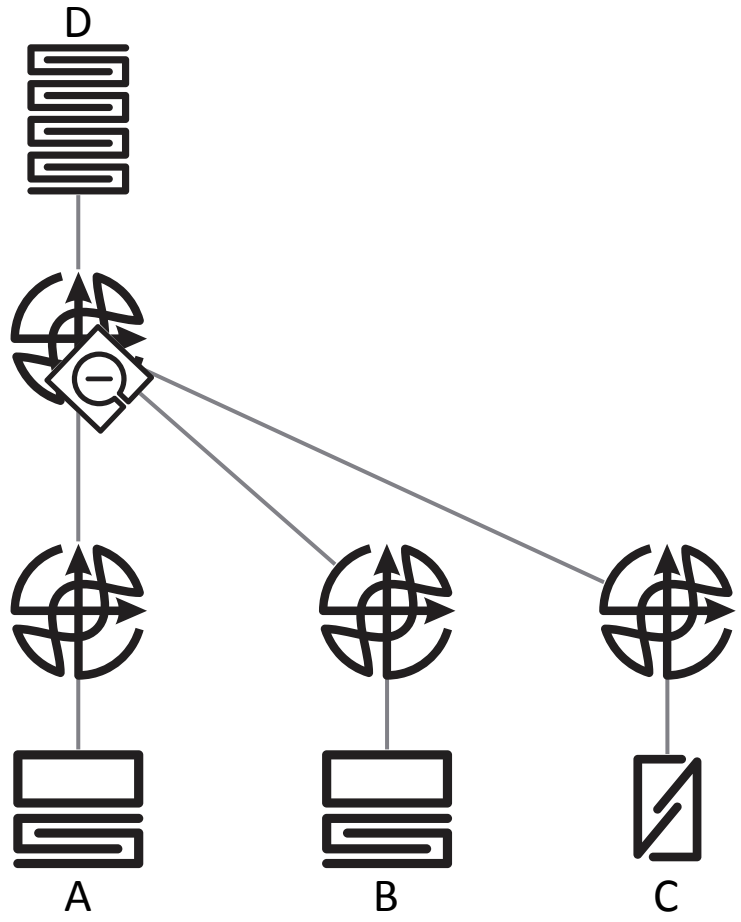
Optimal Forwarding

- At the first hop router?
- We have to manage per edge node
- I can automate these configurations, but…
  - Now I have to manage a new set of tools and processes
- No matter how I slice this, dispersing policy closer to the edge adds complexity



Optimal Forwarding

- At the second hop router?
- Reduces the number of devices to manage
- But…
  - Potentially wastes bandwidth between the first and second hop router
  - Leaves the first hop routers without the packet inspection protection offered at the edge



Optimal Forwarding

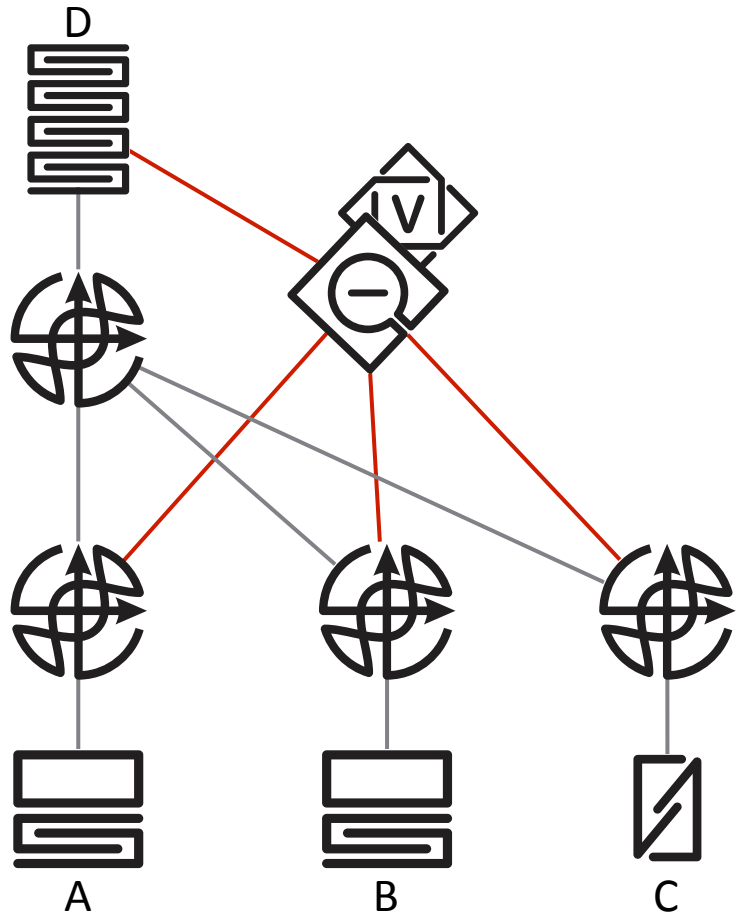
- Dispersing policy increases configuration and management complexity while increasing optimality
- Centralizing policy decreases complexity while decreasing optimality
- This will be true for any service or function that impacts the handling of traffic hop by hop
  - Quality of service, traffic shaping, caching, etc.

Service Chaining

- *I know! I'll just virtualize my services*
  - *Then I can tunnel the traffic service to service starting from where it enters the network!*
- Good try…
  - But you can't fool the demons of complexity that easily…

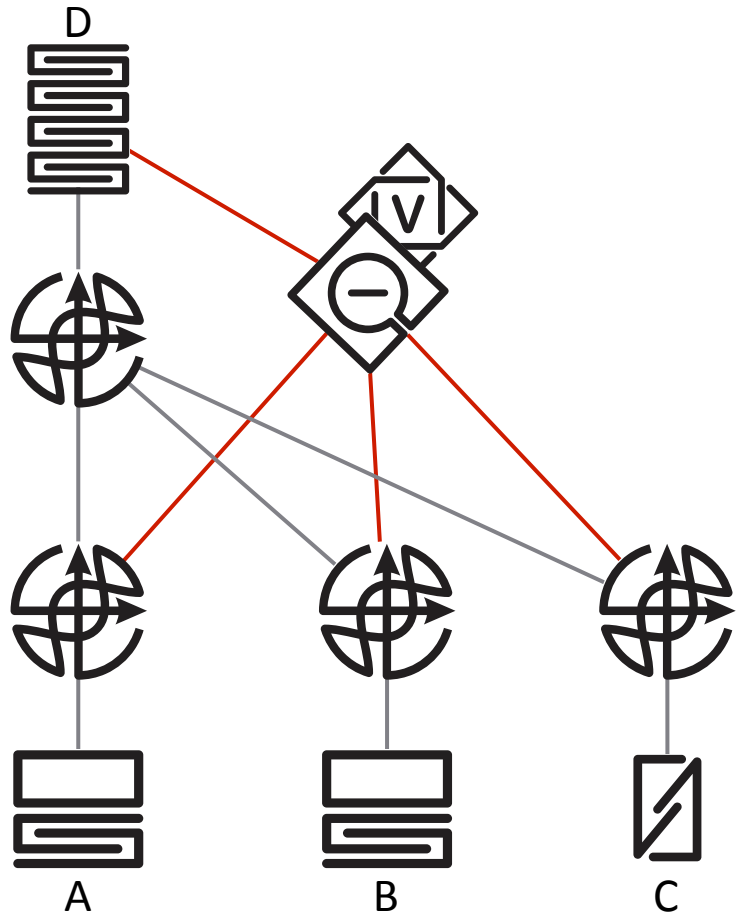
Service Chaining

- Create a new virtual service containing the packet inspection process someplace close to D
- At the network entrance…
  - Look up the destination D
  - Determine the class of service, based on the source A
  - Tunnel the traffic to the virtual packet inspection service



Service Chaining

- We've kept the service logically close to the network edge, while physically centralizing it
- You can bring the policy to your packets, or you can bring the packets to your policy
  - To paraphrase Yaakov's rule…

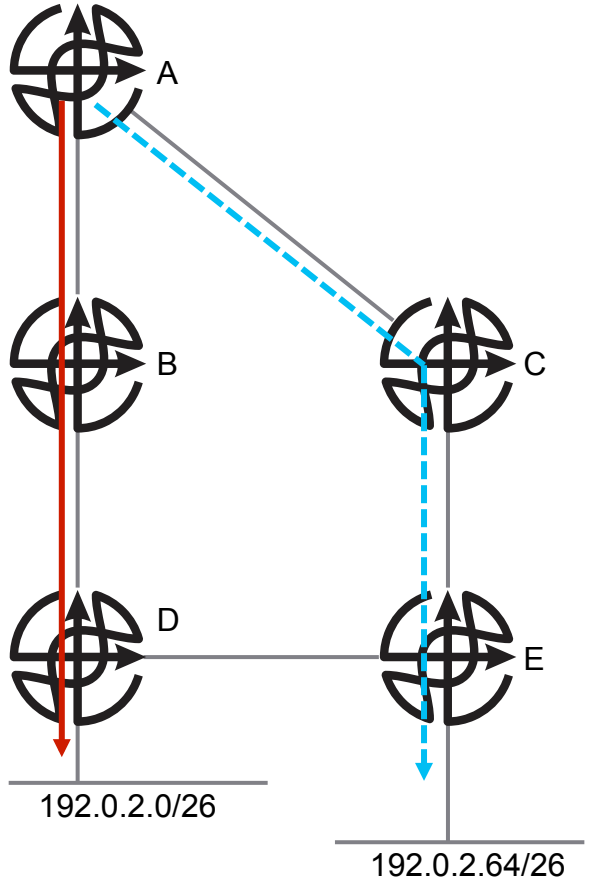


Service Chaining

- We've still added complexity
  - The policy about which packets to put in which tunnels to chain to which services must be programmed in at the edge devices
- And we've still reduced optimality
  - Traffic must be tunneled through the network
    - Potentially wasting bandwidth for packets that will be dropped at some future policy point
    - Tunnels must be configured and maintained
  - Managing quality of service becomes more complex
  - The length of the real path of the packets has increased

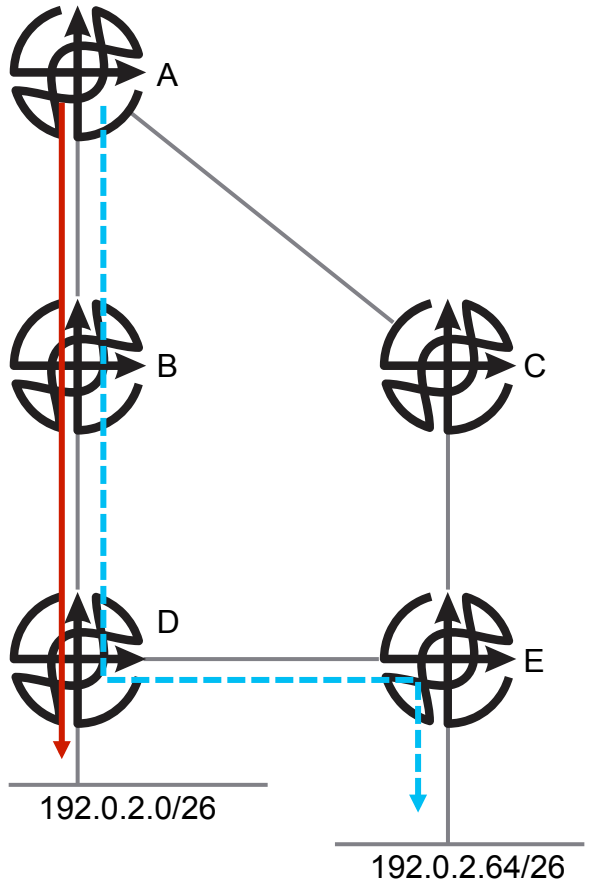
Aggregation/Stretch

- If B and C do not aggregate
  - A will have the optimal route to reach both 192.0.2.0/26 and 192.0.2.64/26
- But…
  - A will have more routes in its local routing table
  - A will receive topology state changes for all the links and nodes behind B and C
  - So more routes, more state change visibility, more complexity



Aggregation/Stretch

- Assume A aggregates to 192.0.2.0/24
  - A will choose either A or B for everything within this subnet (ignoring ECMP)
  - Hence A will choose a suboptimal route to either 192.0.2.0/26 or 192.0.2.64/26
- Reduces complexity
  - A has fewer routes in its local table
  - A deals with less state change over time



Aggregation/Stretch

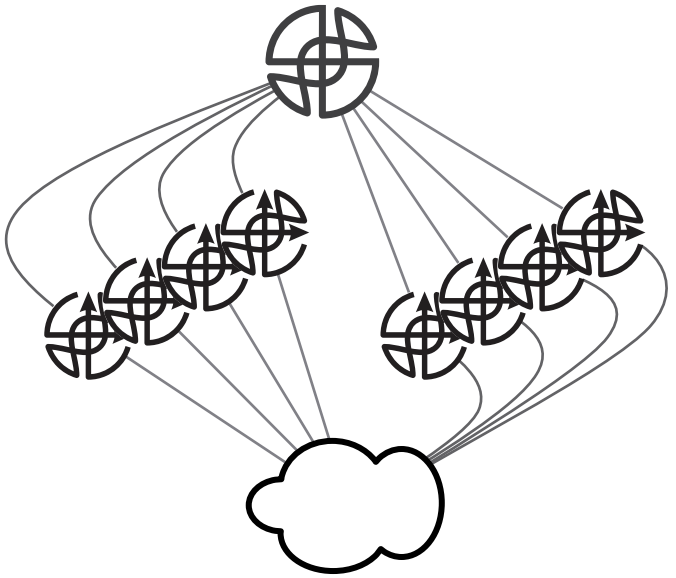
- Aggregation almost always confronts us with the state verses stretch tradeoff
- More state == more optimal paths
- Less state == less optimal paths
  - Or more stretch – the difference between the optimal path through the network and the path the traffic actually takes

Control Plane Centralization

- Let's centralize the entire control plane!
- Won't this be simpler?
  - Policy will be in one place
  - Easier design
    - No thinking through aggregation, etc.
    - Just install the routes where they need to be in real time
  - Can dynamically interact with the control plane in real time
    - Applications can tell the control plane when new paths/etc. are needed
- Sounds neat
  - But… has anyone read RFC1925 recently?
  - *It is always possible to agglutinate multiple separate problems into a single complex interdependent solution. In most cases this is a bad idea.*

Control Plane Centralization

- Complexity Points
  - North/South interface
    - This isn't as simple as it sounds
    - Particularly as there is a "kitchen sink" tendency in these things
  - Resilience
    - The controller is a single point of failure
    - This has to be mitigated somehow…
  - Fast convergence
    - We can always precompute and install alternate paths
    - But double failures and rapidly changing local conditions can stress the system, possibly causing a control plane failure
- Maybe we need a new rule of thumb…
  - Distribute where you can, centralize where you must…



The End!