Declarative Languages in Routing

Fred Baker

"I think you should be more explicit here in step two."
My view on Declarative Languages

- I think that industry will use any tool that makes sense to it in solving its problems.
  
  To apply that rule, we need to understand what problems Industry is trying to solve.

- Let me give you two service provider problems I think make sense to solve
  
  Routing a class of traffic
  Precise load sharing between ISPs

- Then let’s talk about practical software engineering issues
  
  What are the important parts of the problem
  What parts do declarative languages solve?
Cisco forwarding paradigms

- Cisco lookup paradigms:
  - Normal destination-address
    - Standard internet behavior
  - Multi-topology Routing
    - Observe DSCP or other fields to select a route table
  - Policy-based Routing
    - Observe access list to alter/override routing
  - Virtual Route Forwarding
    - Separate route table (FIB) for a set of interfaces

- Issues Cisco is trying to address:
  - Overlapping address spaces (usually RFC 1918 addresses)
  - Variations on the “fish problem”:
    - Customers with different contracts requiring separate routing
    - Traffic engineering
    - Other engineering issues
Routing a class of traffic

- Imagine that instead of “routing toward a destination address” we “routed a class of traffic”

  “Class of traffic” is specified by a classifier

  \{\text{<destination> is in <prefix>}\} \text{ is a simple example of a classifier}

  \text{<next hop> = } F(\{\text{<dest>, <source>, <DSCP>}\}, \text{ <environment}>)?

  \{\text{set of Next hops}\} \text{ is a function of the destination address, source address, and DSCP, potentially other attributes such as local ingress interface ID or recent throughput rate}

  \text{Destination and source addresses may be further mapped to destination, source, previous, or subsequent AS in the process of route calculation}

  \text{Traffic matching classifier is handed to a next hop router or to an interface/tunnel}

  \text{For example, traffic to a remote AS might use a specific subsequent AS, and the next hop router would instantiate a route to that subsequent AS}
Requirements of a routing algorithm for classes of traffic

- Must unambiguously identify traffic classes
  - Must provide a way to disambiguate overlapping classes (“longest match first” and “Multi-exit Discrimination” being examples of such a rule)
  - How about “from this source” or “of this traffic class”?

- Must provably calculate stable routes for every class for which it has information

- Must have a way to accurately select serviceable routes that can be described using some policy. “Least Metric” is an example; so is “greatest unused bandwidth”
**Rate-offload load sharing**

- **Scenario:**
  
  "Brazilian" ISP has large but finite bandwidth to/from its upstreams
  
  Changing this is hard, expensive, and takes time
  
  Total downstream traffic approximates downstream capacity
  
  Current solution:
  
  use BGP traffic engineering to frequently move traffic between downstream links in small units
  
  - **Issues:**
    
    frequent flaps, continuous engineering effort, tempers, expense, route table expansion
  
  - **Issues relevant to any ISP:**
    
    Upstream would like any ingress to send to an egress in
    
    the same POP when possible,
    
    Least expensive POP in any event
    
    Upstream would prefer to drop traffic at ingress if it has to drop at all
    
    Downstream would like to receive all traffic relevant to its customers
    
    Prefers load sharing to dropping of traffic to it
    
    Might write that into a contract and pay for it rather than spend money on BGP traffic engineering
Suppose, just for fun, that…

- …we had each ingress send “the right” proportion of its traffic to each egress?
  
  Egresses tell ingresses what capacity they can dedicate to their traffic
  
  Ingresses send to each egress up to the capacity it can carry by some rule
  
  \[\text{First } <\text{amount}> \text{ goes to } <\text{most preferred egress}>\]
  
  \[\text{Next } <\text{amount}> \text{ goes to } <\text{next egress}>\]
  
  …

- If an ingress can’t find a way to route its traffic, it drops it before expending ISP resources on it
  
  Sort of “input queuing for a service provider network”
Structure of a routing application

- What do declarative languages solve?
  Declarative languages solve the easy part: “how do I apply the rules to determine my routes”
  They do not address the issues of maintaining the routing database, neighbor relationships etc – that is all assumed as an input
  They ask us to fit very different software environments together to save a few lines of code

- For existing protocols, router manufacturers already have implementations that their users trust
  What is their motivation to change them?
So – declarative languages have their best use solving advanced problems

- OSPF, IS-IS, or BGP are interesting proofs of concept
  We have solutions there, and they work
  But OSPF, IS-IS, and BGP don’t solve our advanced problems
  So the fact that you can rewrite their route calculation subroutines isn’t impressive

- Rather, use the technology to address problems that
  We don’t have solutions for
  But we really wish we did
  And figure out where this fits among the rest of the software engineering puzzle

- I think that industry will use any tool that makes sense to it in solving its problems