Metarouting

Timothy G. Griffin
Computer Laboratory
University of Cambridge
Cambridge, UK
timothy.griffin@cl.cam.ac.uk

João Luís Sobrinho
Instituto de Telecomunicações
Instituto Superior Técnico
Lisbon, Portugal
joao.sobrinho@lx.it.pt

ABSTRACT

There is a shortage of routing protocols that meet the needs of network engineers. This has led to BGP being pressed into service as an IGP, despite its lack of convergence guarantees. The development, standardization, and deployment of routing protocols, or even minor changes to existing protocols, are very difficult tasks. We present an approach called Metarouting that defines routing protocols using a high-level and declarative language. Once an interpreter for a metarouting language is implemented on a router, a network operator would have the freedom to implement and use any routing protocol definable in the language. We enforce a clean separation of protocol mechanisms (link-state, path-vector, adjacency maintenance, and so on) from routing policy (how routes are described and compared). The Routing Algebra framework of Sobrinho [25] is used as the theoretical basis for routing policy languages. We define the Routing Algebra Meta-Language (RAML) that allows for the construction of a large family of routing algebras and has the key property that correctness conditions — guarantees of convergence with respect to the chosen mechanisms — can be derived automatically for each expression defining a new routing algebra.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Protocols, Internetworking*

General Terms

Design, Theory, Languages

Keywords

Routing Protocols, Path Algebras, Algebraic Routing

SIGCOMM'05, August 21–26, 2005, Philadelphia, Pennsylvania, USA.

1. GOOD ROUTING PROTOCOLS ARE HARD TO FIND

Routers today come with a fixed number of routing protocols. Network operators and engineers must solve their routing and connectivity problems as best they can with this small set of tools. For IP unicast routing — to which we restrict our attention in this paper — this means that their solutions must use only static routing together with the standardized dynamic routing protocols RIP, OSPF, IS-IS, and BGP [13, 1, 21, 11], or the proprietary EIGRP [23] of Cisco Systems. Although BGP was developed as an inter-domain routing protocol, it is currently being used by many large enterprises as an intra-domain routing protocol (this should not be confused with using internal BGP, or IBGP, which is an essential component of *inter-domain* routing).

Many researchers may be shocked by the idea of using an EGP as an IGP. Network operators are a much more pragmatic group — they have problems to solve *now* and they simply do their best with the tools available. In fact, several recent books on BGP provide advice on how best to "break the rules" and use BGP for intra-domain routing. Chapter 5 of [28] and Chapter 3 of [27] both explain the rationale for this approach and present configuration options to address particular kinds of network challenges. BGP is useful as an IGP because it provides more hierarchical structure and enables the implementation of well-defined administrative boundaries — often essential in geographically dispersed and administratively heterogeneous networks. Of equal importance is the fact that traditional IGPs are all based on *shortest paths* with severe limitations in terms of policy control over routing. For example, with shortest path routing it is very difficult to implement different policies for different destinations.

In short, BGP is being used as an IGP not because it is ideal, but because it is available, it has expressive policy control mechanisms, it can be used to implement administrative boundaries, and it scales well. An unstated reason is related to the difficulty of developing and deploying new routing protocols, or even minor modifications to existing protocols. Beyond the demanding standardization process, the fact remains that it is very difficult to develop well-behaved protocols with the sophisticated policy control required by intra-domain routing in many large enterprises.

This is not a positive development in many regards, since BGP has no convergence guarantees [15]. Furthermore, when BGP is used as an IGP, routing policies have fewer constraints than inter-domain routing where the standard "prefer customer routes over peer routes over provider routes"

policies provides at least partial protection against protocol divergence [6]. Beyond protocol divergence, policy interaction in BGP can result in multiple stable solutions, some intended by policy writers while others are not, and when a routing system becomes *wedged* in an unwanted routing solution it may be very difficult to debug [10].

In this paper we propose a new approach to the definition and deployment of routing protocols called *metarouting*. A metarouting specification defines a routing protocol in a high-level and declarative fashion. Routers need only implement an interpreter (or compiler) for a routing metalanguage in order to run any protocol so specified. Metarouting allows any network operator to specify a new routing protocol and then to use it.

Our approach is based on four ideas. First, we clearly separate protocol mechanisms (link-state, path-vector, hard- or soft-state, adjacency maintenance) from routing policy (how routes are described and compared). Second, the theoretical framework of metarouting is to be found in the long tradition of *path algebras* (see [7, 2, 19] for several examples). That is, the means of describing routes and comparing route preference is captured in algebraic structures having rigorously defined semantics. In this paper we will adopt the *Routing Algebra* framework of Sobrinho [25] as our basic algebraic model. The reason for this is that this algebraic model was developed specifically to address rich policy control as found in BGP. Third, a key novel component of metarouting is the use of a language for defining new and more complex algebras from simpler ones. We present one language for the routing algebras of Sobrinho, called the Routing Algebra Meta-Language (RAML). RAML is a collection of simple base algebras together with a set of operators that take algebras as arguments and return new algebras. In this way, RAML can be used to define a large family of routing algebras. Fourth, RAML is designed so that correctness conditions can be automatically derived for each expression defining a new algebra. For each base algebra certain monotonicity properties are shown to hold. Then each algebraic operator is associated with rules that describe how it preserves the monotonicity properties of its argument algebras. In this way monotonicity properties are easy to derive for any RAML expression.

Metarouting can be viewed within a larger effort attempting to *disaggregate* and *standardize* the components of routing software and hardware, which the vendors have typically built as monolithic systems with many proprietary implementation details and interfaces. This work has included kernel design [3], modular implementation data-plane elements in Click [14], modular routing software as with FIRE [22] and XORP [12], and efforts to standardize the interfaces and protocols for low-level forwarding units in the FORCES working group [5] of the IETF. A missing aspect has been how to deal with the complex policy component of routing protocols in a generic yet high-level manner — and this is what metarouting is attempting to provide.

We do not imagine that every network operator would in fact want to define their own protocols. We imagine that metarouting could eventually enable a natural division of labor between the IETF and the network operator community — metarouting itself could be standardized within the IETF, while metarouting specifications of routing protocols could be developed and standardized within the operator community.

Section 2 describes the decomposition of a routing protocol into mechanism and policy components, reviews the routing algebras of Sobrinho [25], and describes algebraic properties of routing algebras that guarantee convergence with a specific routing algorithm. Section 3 presents the Routing Algebra Meta-Language, RAML, and develops monotonicity preservation properties of each RAML construct. For readability and space reasons, all proofs have been eliminated. In Section 4 we show examples of using metarouting to develop and implement a new IGP. Developing new routing protocols will never be a trivial task, but we feel that metarouting will reduce the associated effort by several orders of magnitude. Section 5 presents a metarouting model of BGP, and considers how this approach might aid in improving this protocol. This section illustrates how RAML can aid researchers by providing a framework for the analysis of routing protocols. Section 6 defines *label modalities* that describe how abstract link labels are actually constructed from the information in router configurations on either side of a routing adjacency. Modalities represent a rug under which we sweep some of the protocol details that are not important from a purely theoretical perspective, but are important from the perspective of a network operator configuring a network of routers. Section 7 concludes with a discussion of directions for future research.

2. WHAT IS A ROUTING PROTOCOL?

At the highest level, we can decompose most routing protocols into two components — mechanism and policy. By policy we mean the information that describes the characteristics of a route, the method of comparing route characteristics to determine route preference, and the method in which local policy is applied to routes, potentially changing a route's characteristics or limiting the scope of its propagation. By mechanism we mean how routing messages are exchanged, how routing adjacencies are established and maintained, and what type of route selection algorithms are used to select best routes. Route selection algorithms rely on the policy component to determine route preferences.

It might seem that "route selection algorithm" and "the method of comparing route characteristics to determine route preference" refer to the same thing. However, they are not for a large class of routing protocols. A simple example may help clarify this point. Shortest-path routing attaches weights to links and the important characteristic of a route is the sum of these link weights along the path it represents. We can talk about one route being more preferred than another when it is associated with a lower cost path. Now, there are several route selection algorithms that use this method of preference to compute best routes — link information flooding combined with Dijkstra's algorithm and distributed Bellman-Ford are the most well known examples, and of course these are the algorithms associated with link-state and distance-vector routing protocols.

What allows us to generalize this picture is an *algebraic approach to routing* in the tradition of Gondran, Minoux, and Carré [7, 2, 19]. In this paper we use the routing algebras as defined by Sobrinho [25], which are reviewed in detail in Section 2.1. A routing algebra comes with a set of *signatures* that describes characteristics of a route. It also defines the method of determining route preference based only on route signatures. Finally, a routing algebra generalizes the notion of a link weight to a *policy label* (referred to

simply as a "label"), and it defines how a policy label and a signature combine to form a new signature.

2.1 Routing Algebras

We now provide a brief overview of routing algebras as developed by Sobrinho [24, 25]. Routing algebras are best understood as a generalization of shortest path routing. This is illustrated in Figure 1(a) for the traditional shortest path scenario. Node v has a path to a route originated by node w, indicated by the dotted line, and the length of this path has been computed to be m (this path may involve multiple nodes). Node v has a neighbor u, and there is an arc from node v to u with weight n. The composition of the $(v,\ u)$ arc with the w to v path results in a path from w to u of length $n + m$.

We use the convention that the arc's orientation points in the same direction as the flow of routing information in a path-vector protocol. Note that data traffic associated with a route travels in the opposite direction, from u to w.

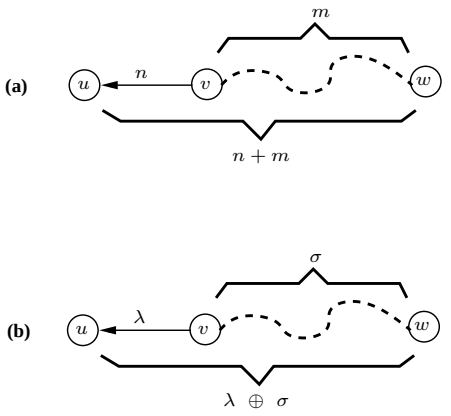

Figure 1: How path lengths are computed in the standard shortest path setting (a), and how path signatures are computed in the routing algebra setting (b).

This basic picture is generalized by routing algebras and is illustrated in Figure 1 (b). A *routing algebra* A is a tuple

$$A = \langle \Sigma, \preceq,\ L,\ \oplus,\ \mathcal{O} \rangle,$$

where Σ is a set of *signatures* for describing paths, $\preceq$ is a *preference relation* over signatures, L is a set of *labels*, $\oplus$ is a label application function that maps $L \times \Sigma$ to Σ, and $\mathcal{O}$ is an *origination set* describing the signatures that can be associated with originated routes.

A preference relation (commonly used in economics, see for example [17]), conforms to two rules,

(completeness) for each $x,\ y \in \Sigma$, we have either $x \preceq y$ or $y \preceq x$ (or both),

(transitivity) for each $x,\ y,\ z \in \Sigma$, if $x \preceq y$ and $y \preceq z$, then $x \preceq z$.

If $x \preceq y$ we say that x *is weakly preferred to* y. If $x \preceq y$ but $y \preceq x$ does not hold, then we write $x \prec y$ and say that x *is strictly preferred to* y. If $x \preceq y$ and $y \preceq x$, then we write $x \sim y$ and say that x *and* y *are equally preferred*. Note that $x \sim y$ does not mean that $x = y$. This fact is important for both the expressiveness of routing algebras and for the modeling of equal-cost multipath routing. For example, consider the simple case where Σ represents sequences of integers (perhaps router IDs or ASNs) and $x \preceq y$ holds if and only if the length of x is less than or equal to the length of y. Then $(2,\ 3, 4) \prec (7,\ 1,\ 2,\ 5)$ and $(2,\ 3, 4) \sim (7,\ 1,\ 2)$ yet $(2,\ 3, 4) \neq (7,\ 1,\ 2)$. This is exactly the kind of comparison that BGP uses on the ASPATH attribute.

Note that we have reversed the preference order with respect to the conventions found in most economics texts. We do this because we will be using preference to *minimize* path cost rather than to *maximize* some benefit. Our routing algebra notation differs somewhat from that presented in [24, 25] — we use a preference relation instead of its implementation in terms of utility functions. Following the original notation we would say that Σ comes with an associated set of *weights* W totally ordered with $\leq$ and a *ranking function* f that maps Σ into W. In the notation above, $\sigma_1 \preceq \sigma_2$ means that $f(\sigma_1) \leq f(\sigma_2)$.

Optionally, an algebra may come with a special signature, $\phi \in \Sigma$, which is associated with *prohibited paths* — paths with signature ϕ cannot be used for forwarding and are not propagated by routing protocols. We insist that for all $\sigma \neq \phi$ we have $\sigma \prec \phi$.

The job of a dynamic routing protocol is to compute *routes*, and for us a route will have the form

$$r = \langle p,\ nh,\ \sigma \rangle,$$

where p represents a set of destination addresses (a prefix), nh is the next-hop address, and σ is the signature describing the characteristics of this route. Such routes could represent static routes, or be computed by a routing protocol.

In Figure 1 (b), the signature $\sigma \in \Sigma$ describes the path from node v to the originating node w, and $\lambda \in L$ describes the policy applied on the arc from node v to node u. The signature describing the path from node w to u is then $\lambda \oplus \sigma$. Labels and signatures may contain rather complex objects, and it is not required that $L = \Sigma$.

For route origination, we will define a set of *origination signatures*, $\mathcal{O} \subseteq \Sigma$, to constrain the signatures that can be legally attached to routes that are injected into the protocol, either from static routes or from other protocols. For example, in BGP originated routes must have an empty ASPATH, as seen by the originating router.

2.2 Convergence Guarantees

The basic notion of correctness for routing protocols can be informally stated as follows. Once all changes have ceased in a network, a routing protocol should eventually determine stable forwarding tables that implement loop-free paths between every pair of endpoints that are allowed connectivity by policy.

In the algebraic approach to routing, correctness is ensured with a clean separation of concerns. First, certain algebraic properties are identified for routing algebras. Second, generic algorithms are developed and proved correct for *any* algebra having the algebraic properties required by the algorithm.

For vectoring algorithms (generalizations of distributed Bellman-Ford), Sobrinho [25] showed that the important algebraic property is *strict monotonicity* (SM):

(SM) For all $\sigma \in \Sigma - \{\phi\}$, and for all $\lambda \in L$, $\sigma \prec \lambda \oplus \sigma$.

A vectoring algorithm that is using an SM algebra will always be correct. The SM property can also be used to show that there can be no forwarding loops in the resulting forwarding paths.

The RAML presented in Section 3 is designed to make it easy to derive complex SM algebras. In order to do this we need a slightly weaker property, called *monotonicity* (M):

(M) For all $\sigma \in \Sigma$, and for all $\lambda \in L$, $\sigma \preceq \lambda \oplus \sigma$.

Note that SM implies M.

Protocols such as IS-IS and OSPF are not based on vectoring but use link-state approaches that rely on several distinct components. First, a link-state flooding mechanism is used to distribute each router's local information to all other routers in the link-state routing domain. Second, an algorithm is used locally on each router that computes best paths in the network, as modelled by a weighted graph, and uses these paths to determine next-hops for each route. Typically, some version of Dijksta's algorithm [4] is employed. Third, forwarding paths are constructed by the concatenation of the next hops determined independently at each node.

It is possible to generalize link state flooding and Dijkstra's algorithm to an arbitrary algebraic context [19, 24]. However, requirements for correctness are more restrictive. We require the associativity of $\oplus$ (which in turn requires that $L = \Sigma$), and that the algebra be *isotonic* (I):

(I) For all $\sigma_1,\ \sigma_2,\ \sigma_3 \in \Sigma$, if $\sigma_1 \preceq \sigma_2$, then $\sigma_3 \oplus \sigma_1 \preceq \sigma_3 \oplus \sigma_2$ and $\sigma_1 \oplus \sigma_3 \preceq \sigma_2 \oplus \sigma_3$.

In addition, SM for such algebras must include the SM defined above (left-SM) as well as a right-SM rule. When constructing new algebras it turns out that these additional constraints are very difficult to preserve.

It is important to note that we can still call a mechanism "link-state" even when the local algorithm bears no relation to Dijkstra's algorithm. In particular, if an algebra is only SM, then one could use a link-state approach with a local algorithm that essentially *simulates* vectoring on a local model of the network. This is not as strange as it might seem at first glance. In the case that the routing domain is not too large, it may actually be a reasonable, especially if fast convergence *and* complex policy control are *both required*. We will call this approach *Local Path-Vector Simulation* (LPVS). Table 1 indicates when an algebra is correctly associated with a given algorithm.

	SM	I	assoc. $\oplus$
vectoring	$\surd$		
link-state with Generalized Dijkstra	$\surd$	$\surd$	$\surd$
link-state with LPVS	$\surd$		

Table 1: Correctness for various algebra/algorithm combinations.

2.3 Base Algebras

Table 2 presents a collection of simple routing algebras, together with their monotonicity property. Each algebra is now explained in turn.

Algebra	Description	Properties
ADD($n,\ m$)	int addition	SM (if $1 \le n \le m$)
MULT($n,\ m$)	int product	M (if $1 \le n \le m$)
MULT$_r$($n,\ m$)	real product	–
MAX(n)	maximum	M
MIN(n)	minimum	–
LP(n)	local preference	–
OP(n)	origin preference	M
SEQ($n,\ m$)	sequences	SM
SIMSEQ($n,\ m$)	simple sequences	SM
TAGS(t)	route tags	M

Table 2: Basic Routing Algebras.

For integers n and m, the routing algebra ADD($n,\ m$) represents addition in the range n to m. It has integer sets $L = \mathcal{O} = \{n,\ \ldots,\ m\}$ and $\Sigma = \{n,\ \ldots,\ m\} \cup \{\phi\}$, where $\preceq$ is taken to be the standard order $\le$ on integers, extended to make ϕ the least preferred. The operator $\oplus$ is defined as normal addition, except for values that are *out of range*:

$$i \oplus j = \begin{cases} \phi & \text{if } (i+j) \notin \{n, \ldots, m\}, \\ i+j & \text{otherwise.} \end{cases}$$

For example, here is the $\oplus$ table for ADD(1, 5):

	$\oplus$	1	2	3	4	5	ϕ
L	1	2	3	4	5	ϕ	ϕ
	2	3	4	5	ϕ	ϕ	ϕ
	3	4	5	ϕ	ϕ	ϕ	ϕ
	4	5	ϕ	ϕ	ϕ	ϕ	ϕ
	5	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

(Σ)

This table conforms to a convention used throughout the paper — rows are associated with labels displayed in the leftmost column, while columns are associated with signatures, displayed along the top row. The signatures are presented with preference decreasing from left to right.

MIN(3)

$\oplus$	1	2	3
1	1	1	1
2	1	2	2
3	1	2	3

MAX(3)

$\oplus$	1	2	3
1	1	2	3
2	2	2	3
3	3	3	3

LP(3)

$\oplus$	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

OP(3)

$\oplus$	1	2	3
κ	1	2	3

Figure 2: The $\oplus$ tables of several simple routing algebras.

The algebra MULT($n,\ m$) is are defined in a similar manner for integer multiplication. The algebra MULT$_r$($n,\ m$) represents real multiplication (at some fixed precision), for reals between n and m. The algebras MIN(n) and MAX(n) both have $L = \Sigma = \mathcal{O} = \{1,\ \ldots,\ n\}$, with the preference relation being the standard order $\le$ on integers. The operations $i \oplus j$ are defined to be the minimum and maximum,

respectively, of $\{i,\ j\}$. Figure 2 presents the $\oplus$ tables for MIN(3) and MAX(3).

The algebra LP(n) (*local preference*) has $L = \Sigma = \mathcal{O} = \{1,\ \ldots,\ n\}$ and the $\oplus$ operator is defined as $i \oplus j = i$. That is, the last link in a path determines the total path weight. The dual algebra, OP(n) (*origin preference*), has $L = \{\kappa\}$, $\Sigma = \mathcal{O} = \{1,\ \ldots,\ n\}$, and $\oplus$ operator is defined as $\kappa \oplus j = j$. That is, signatures can only be copied with κ and the originating node determines the total path weight — the only way signatures can be attached to routes is at origination. The $\oplus$ tables for LP(3) and OP(3) are also presented in Figure 2.

We will also use several base algebras that do not lend themselves well to tabular presentation due to the large number of entries. Signatures in the routing algebra SEQ($n,\ m$) are finite sequence σ over the integers in $\{0, \ldots, n\}$, whose length $\mid \sigma \mid$ is at most m. The preference relation is defined as $\sigma_1 \preceq \sigma_2 \stackrel{\text{def}}{=} \mid \sigma_1 \mid \leq \mid \sigma_2 \mid$. The $\oplus$ operation is defined as

$$i \oplus \sigma = \begin{cases} \phi & \text{if } m = \mid \sigma \mid, \\ i :: \sigma & \text{otherwise,} \end{cases}$$

where $i :: \sigma$ denotes adding i to the head of the sequence σ. The algebra of simple sequences, SIMSEQ($n,\ m$), is defined in the same manner, except that $\oplus$ is defined as

$$i \oplus \sigma = \begin{cases} \phi & \text{if } m = \mid \sigma \mid \text{ or } i \in \sigma, \\ i :: \sigma & \text{otherwise.} \end{cases}$$

In both cases the origination set is defined as $\mathcal{O} = \{()\}$, where () represents the empty sequence.

Finally, if t is some type, such as INT or STRING, then the signatures of the *tags algebra*, TAGS(t), represents all finite sets of objects of type t. All such sets are given *equal preference*. The $\oplus$ operation allows for insertion and deletion of elements and copying:

$\oplus$	σ
$i(\sigma_1)$	$\sigma \cup \sigma_1$
$d(\sigma_1)$	$\sigma - \sigma_1$
κ	σ

We will see later that tags are very useful for implementing complex routing policies.

3. A ROUTING ALGEBRA METALANGUAGE

A *metarouting language* is any language that allows us to define routing protocol, RP, as

$$RP = \langle A,\ \mathcal{M},\ LM \rangle,$$

where A is a routing algebra, $\mathcal{M}$ is a set of mechanisms that can be associated with a routing adjacency (multiple mechanisms may be used for the same protocol, and LM is a set of *label modalities* that are described in Section 6).

Constructing new routing algebras by hand, especially complex ones, can be a difficult and tedious task. This is even more true when we are required to prove monotonicity conditions. To address this, we present a Routing Algebra Meta-Language (RAML) for the specification of new routing algebras. RAML represents a compromise — every expressions in RAML represents a routing algebra, but it is certainly not the case that any routing algebra can be expressed in RAML. In compensation, monotonicity conditions can be automatically derived for RAML specifications, much like types in many programming languages. That is, no tedious proofs are required of the protocol designer using RAML. Such a formalism could take many forms, but the one presented here represents our own attempt to strike a balance between mathematical simplicity and usefulness.

If X and Y are sets, we use the notation $X \uplus Y$ to denote their *disjoint union*, which can be defined as $\{\langle 0,\ x \rangle \mid x \in X\} \cup \{\langle 1,\ y \rangle \mid y \in Y\}$.

3.1 Lexical Product, $A \otimes B$

We start with binary operations. Suppose we are given two routing algebras,

$$A = \langle \Sigma_A,\ \preceq_A,\ L_A,\ \oplus_A,\ \mathcal{O}_A \rangle, \quad B = \langle \Sigma_B,\ \preceq_B,\ L_B,\ \oplus_B,\ \mathcal{O}_B \rangle,$$

and we want to define binary operators $\bullet$ for constructing new routing algebra $A \bullet B$,

$$A \bullet B = \langle \Sigma,\ \preceq,\ L,\ \oplus,\ \mathcal{O} \rangle,$$

We would like the definition of each operator to be fairly simple and natural.

One natural approach is to take Σ as the product $\Sigma_A \times \Sigma_B$ and define $\preceq$ as the lexicographic preference relation:

$$\langle \sigma_A,\ \sigma_B \rangle \preceq \langle \beta_A,\ \beta_B \rangle \stackrel{\text{def}}{=} \\ \sigma_A \prec_A \beta_A \text{ or } (\sigma_A \sim_A \beta_A \text{ and } \sigma_B \preceq_B \beta_B).$$

A bit of care must be taken if either Σ_A or Σ_B contains the prohibited signature ϕ. In this case we define Σ as $((\Sigma_A - \{\phi\}) \times (\Sigma_B - \{\phi\})) \cup \{\phi\}$, and extend the definition of $\preceq$ so that $\langle \sigma_A,\ \sigma_B \rangle \prec \phi$ for all $\langle \sigma_A,\ \sigma_B \rangle \in \Sigma$.

One way to apply labels to product signatures is to do it pair-wise. Define $\mathcal{O}$ as $\mathcal{O}_A \times \mathcal{O}_B$, L as $L_A \times L_B$, and $\oplus$ as

$\oplus$	$\langle \sigma_A,\ \sigma_B \rangle$
$\langle \lambda_A,\ \lambda_B \rangle$	$\langle \lambda_A \oplus_A \sigma_A,\ \lambda_B \oplus_B \sigma_B \rangle$,

where $\lambda_i \in L_i$ and $\sigma_i \in \Sigma_i$. However, if either $\lambda_A \oplus_A \sigma_A$ or $\lambda_B \oplus_B \sigma_B$ is equal to ϕ, then

$$\langle \lambda_A,\ \lambda_B \rangle \oplus \langle \sigma_A,\ \sigma_B \rangle = \phi.$$

In addition, it is always the case that $\langle \lambda_A,\ \lambda_B \rangle \oplus \phi = \phi$.

Product algebras are very useful for routing protocols with multiple routing metrics. For example, we can think of the route selection of BGP as a lexicographic comparison of multiple attributes (see Section 5 for more details). OSPF provides another example that may not be so obvious. At first glance, it may seem that OSPF is a simple protocol needing only an algebra of the form ADD(1, m). However, careful reading of the protocol specification [20] reveals that in fact it is using a lexicographic ordering. An OSPF signature can be modeled as a pair, $\langle \alpha, d \rangle$, where α contains area information and d represents distance. Route preference must be defined so that intra-area routes are preferred over inter-area routes, no matter what the values of route distance, and this can be accomplished with lexicographic preference. (For more discussion of OSPF, see Section 7.)

The binary lexical product naturally generalizes to an n-ary lexical product,

$$\otimes(A_1,\ A_2,\ \ldots,\ A_n).$$

It is often useful to have some means of naming the individual components of a products signature for ease of notation when it comes to writing conditional policy labels described below. For this we introduce the unary operator $\mathbf{a} : A$, which

produces a routing algebra which is exactly like A, except that each signature is now a pair $\langle \mathbf{a},\ \sigma \rangle$, usually written as $\mathbf{a} : \sigma$. Now a fully labeled n-ary product can then be written as

$$\otimes(\mathbf{a}_1 : A_1,\ \mathbf{a}_2 : A_2,\ \ldots,\ \mathbf{a}_n : A_n),$$

where the $\mathbf{a_i}$ are unique labels associated with the sub-algebras. All (non-ϕ) signatures then have the form

$$\langle \mathbf{a}_1 : \sigma_1,\ \mathbf{a}_2 : \sigma_2,\ \ldots,\ \mathbf{a}_n : \sigma_n \rangle.$$

If all attributes of a product are uniquely labeled, then an implementation need not enforce a strict order on the sequence of values in a tuple.

Another useful feature for products, present in BGP, is the ability to have *optional* attributes. There are several reasonable ways of accomplishing this within RAML, each with slightly different semantics. Although optional arguments may seem trivial at first glance, they illustrate well the metarouting approach. Our desire to preserve algebraic properties, such as M and SM, provides a rigorous framework in which to explore alternative definitions. And we can do this in isolation from the distracting complexities — inherent and accidental — of a particular routing protocol. Once we sort out these issues and encode our understanding in well-defined operators, we then can use them repeatedly to construct complex yet mathematically tractable protocols.

First, we must decide how missing values are treated by the preference relation. Suppose we are defining a version of $A \otimes B$ where the first element is optional, and we denote the missing value with the *null signature* $\bot$. We could extend the definition of $\preceq$ on product algebras so that

$$(\bot,\ \sigma_B) \preceq (\beta_A,\ \beta_B) \Leftrightarrow (\sigma_A,\ \sigma_B) \preceq (\bot,\ \beta_B) \Leftrightarrow \sigma_B \preceq_B \beta_B.$$

That is, if one of the first elements is $\bot$, we *ignore* this element and use only the $\preceq_B$ preference relation of B. But there is a problem with this. For any σ_A, β_A, we have $(\sigma_A,\ \sigma_B) \preceq (\bot,\ \sigma_B) \preceq (\sigma_A,\ \sigma_B)$. Therefore $(\bot,\ \sigma_B) \sim (\sigma_A,\ \sigma_B)$. That is, if $\preceq$ is to be a preference relation, it must be that the $\sim$-class of every tuple $(\sigma_A,\ \sigma_B)$ is identical to that of $(\bot,\ \sigma_B)$. This does not seem to be a fruitful approach — the first component serves no purpose!

Rather than modify the definition of $\otimes$, it seems more reasonable to define a *unary* operator $\bot(A)$ that creates a version of A that allows for the null signature $\bot$. It then seems reasonable that the $\oplus$ operation would then be some combination of the following rules:

$\oplus$	$\bot$	σ_A
λ	$\bot$	$\lambda \oplus_A \sigma_A$
$\bot$	$\bot$	$\bot$
$\hat{\sigma}_A$	$\hat{\sigma}_A$	ϕ

The rule $\hat{\sigma}_A \oplus \bot = \hat{\sigma}_A$ allows a signature to be treated as a label, and turns a null signature into $\hat{\sigma}_A$ (so in this case, we need to define L as $(L_A \cup \{\bot\}) \uplus \Sigma$). But how should we define the preference of $\bot$? Since $\bot \oplus \bot = \bot = \lambda \oplus \bot$, the resulting algebra will never be SM, no matter how the preference of $\bot$ is defined. However, suppose that we want $\bot(A)$ to preserve M. Then $\bot \oplus \sigma_A = \bot$ tells us that for all $\sigma \neq \phi$, $\sigma \preceq \bot$. That is, $\bot$ must be a least preferred signature. On the other hand, the rule $\hat{\sigma}_B \oplus \bot = \sigma_B$ tells us that, to preserve M, we need $\bot \preceq \sigma$ for all $\sigma \neq \phi$. That is, in this case $\bot$ must be a most preferred signature.

To solve this issue we define *four* flavors of $\bot(A)$. The algebras $\bot(\text{min},\ A)$ and $\bot(\text{max},\ A)$ both have the $\oplus$ table defined above, but the first takes $\bot$ as a minimal element, while the second takes $\bot$ to be a maximal element. Neither preserves M. Next we define two M-preserving variants, $\bot_p(\text{min},\ A)$, that gives $\bot$ a minimal preference and $\bot_p(\text{max},\ A)$, that gives $\bot$ a maximal preference. Their $\oplus$ tables are defined as

$\bot_p(\text{min},\ A)$

$\oplus$	$\bot$	σ_A
λ	$\bot$	$\lambda \oplus_A \sigma$
σ_B	σ_B	ϕ

$\bot_p(\text{max},\ A)$

$\oplus$	$\bot$	σ_A
λ	$\bot$	$\lambda \oplus_A \sigma$
$\bot$	$\bot$	$\bot$

Note that with $\bot_p(\text{max},\ A)$, once an element is $\bot$, it will stay $\bot$.

3.2 Scoped Product, $A \odot B$

Another way to apply labels to product signature is to do it point-wise:

$\oplus$	$\langle \sigma_A,\ \sigma_B \rangle$
λ_A	$\langle \lambda_A \oplus_A \sigma_A,\ \sigma_B \rangle$
λ_B	$\langle \sigma_A,\ \lambda_B \oplus_B \sigma_B \rangle$

This is mathematically clean, but it does not seem to be very useful for the definition of routing protocols. However, a small modification produces

$\oplus$	$\langle \sigma_A,\ \sigma_B \rangle$
$\langle \lambda_A,\ \hat{\sigma}_B \rangle$	$\langle \lambda_A \oplus_A \sigma_A,\ \hat{\sigma}_B \rangle$
λ_B	$\langle \sigma_A,\ \lambda_B \oplus_B \sigma_B \rangle$

That is, $\langle \lambda_A,\ \hat{\sigma}_B \rangle \oplus \langle \sigma_A,\ \sigma_B \rangle = \langle \lambda_A \oplus_A \sigma_A,\ \hat{\sigma}_B \rangle$ states that λ_A is applied to the first component while the second component is *replaced* by $\hat{\sigma}_B \in \Sigma_B$. The new label set L is defined to be $(L_A \times \mathcal{O}_B) \uplus L_B$. This algebra, called the *scoped product*, has an interesting interpretation — it captures routing with administrative regions and boundaries in the style of BGP. However, it does this in a completely generic fashion, assuming only that algebra A is used *between* administrative entities, while B is used *inside* of each administrative entity.

Figure 3 illustrates this with a simple scenario having two administrative regions (enclosed in dashed lines). Region 1 is made of up of routers 1 and 2 and region 2 is made up of routers 3 and 4. We do not want internal routing information (using signature of B) to be exported beyond region boundaries. We do want external information (using signature of A) to flow into and out of administrative regions. Suppose $\sigma \in \Sigma_A$ and $\beta \in \Sigma_B$. We will use pairs $\langle \sigma,\ \beta \rangle$ as signatures, with lexical preference. Inside of an administrative region, B labels only will be used to change the β component.

For example, in Figure 3, router 1 originates the signature $\langle \sigma_0,\ \beta_0 \rangle$, and passes this to router 2. The policy arc from router 1 to router 2 is labeled with $\lambda_B^1 \in L_B$, producing the signature $\langle \sigma_0,\ \beta_1 \rangle = \langle \sigma_0,\ \lambda_B^1 \oplus_B \beta_0 \rangle$ at router 2. Router 3 is in another administrative region, and so the label on the policy arc from router 2 to router 3 must supply *both* a label $\lambda_A \in L_A$ *and* an initializing value $\beta_2 \in \mathcal{O}_B$ for the B component. This produces the signature $\langle \sigma_1,\ \beta_2 \rangle = \langle \lambda_A \oplus_A \sigma_0,\ \beta_2 \rangle$ at router 3. Finally, the internal link from router 3 to router 4 is labeled with $\lambda_B^2 \in L_B$, producing the signature $\langle \sigma_1,\ \beta_3 \rangle = \langle \sigma_1,\ \lambda_B^2 \oplus_B \beta_2 \rangle$ at router 4.

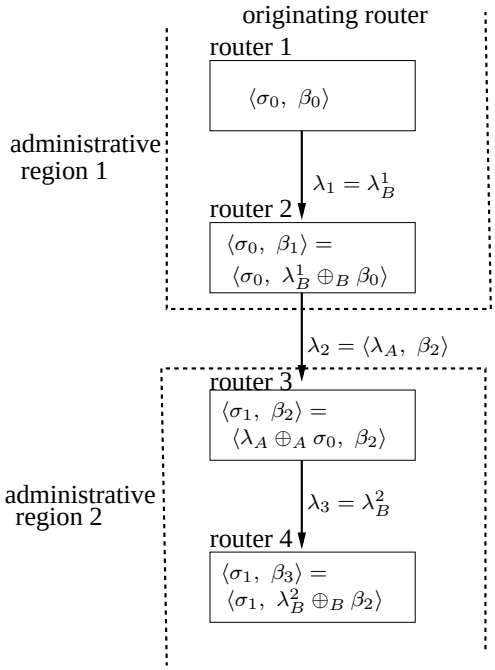

Figure 3: Illustration of the scoped product $A \odot B$.

3.3 Disjunction, $A \lhd B$

Suppose we want to define an operation that allows the use of either A or B or both together. We define a new signature set Σ as $\Sigma_A \uplus \Sigma_B$. Then force every $\sigma_A \in \Sigma_A$ to have a higher preference than every $\sigma_B \in \Sigma_B$. That is, if $\sigma_1, \sigma_2 \in \Sigma_A \uplus \Sigma_B$, then $\sigma_1 \preceq \sigma_2$ if $\sigma_1, \sigma_2 \in \Sigma_A$ and $\sigma_1 \preceq_A \sigma_2$, or if $\sigma_1, \sigma_2 \in \Sigma_B$ and $\sigma_1 \preceq_B \sigma_2$, or if $\sigma_1 \in \Sigma_A$ and $\sigma_2 \in \Sigma_B$. Let t be an *injection function* that maps Σ_A to Σ_B. Define the $\oplus$ operator of $A \lhd_t B$ as

$\oplus$	σ_A	σ_B
λ_A	$\lambda_A \oplus_A \sigma_A$	ϕ
λ_B	ϕ	$\lambda_B \oplus_B \sigma_B$
i	$t(\sigma_A)$	ϕ

We will use $A \lhd B$ to denote disjunction when the last rule is not used.

3.4 Programmatic Labels, PROG(A)

So far labels have been fairly simple constructs. We now show how labels can be extended in a natural way to *programmatic labels*. Suppose that algebra A has labels L_A and operator $\oplus_A$. The signature, preference relation, and origination set of PROG(A) are those of A. The set L and the $\oplus$ operation are defined inductively as follows. First, the set L_A is contained in L, and for all $\lambda_A \in L_A$ and all $\sigma_A \in \Sigma_A$ we have $\lambda_A \oplus_A \sigma_A = \lambda_A \oplus \sigma_A$. Second, the *rejection label* ϕ is in L and $\phi \oplus \sigma = \phi$. Third, if $\lambda_A, \lambda_B \in L$, then the *sequential label* $\lambda = \lambda_A; \lambda_B$ is in L and

$$(\lambda_A; \lambda_B) \oplus \sigma_A = \lambda_A \oplus (\lambda_B \oplus \sigma_A).$$

Fourth, if p is a predicate over a signature set Σ_A and $\lambda_1, \lambda_2 \in L$, then the *conditional label*

$$\lambda = \text{if } p \text{ then } \lambda_1 \text{ else } \lambda_2$$

is in L and

$$(\text{if } p \text{ then } \lambda_1 \text{ else } \lambda_2) \oplus \sigma_A = \begin{cases} \lambda_1 \oplus \sigma_A & \text{if } p(\sigma_A) \\ \lambda_2 \oplus \sigma_A & \text{otherwise} \end{cases}$$

We assume that predicates p are simple boolean formulas constructed over atomic predicates over the domains associated with base algebras. The remarkable thing about PROG(A) is that it preserves both M and SM while greatly increasing policy expressiveness. For example, different "base labels" ($\lambda \in L_A$) can be applied to different routes, based on properties of the associated signatures.

3.5 Monotonicity Preservation

A	B	$A \otimes B$	$A \odot B$	$A \lhd_t B$
M	M	M	–	M
M	SM	SM	–	M
SM	M	SM	M	M
SM	SM	SM	SM	SM
SM	$*$	SM	–	–

Table 3: Binary operators of RAML.

Table 3 presents the preservation properties of the binary operators. The last line of this table means that if A is SM, then $A \otimes B$ is SM, no matter what properties hold for B. This has very important protocol design implications. Figure 4 restates the property preservation as a *design pattern* for n-ary products — if we want such products to be SM, then it is enough to have a sequence of algebras having M, followed by at least one algebra having SM, followed by any routing algebras. Put another way, if you are going to have a well-behaved n-ary product and you have some sub-algebras that do not have any nice properties, then put them "to the right" of well-behaved algebras in the product.

$$\underbrace{\otimes(\overbrace{A,\ \ldots,\ A_{k-1}}^{\text{all M or SM}},\ \overbrace{A_k}^{\text{SM}},\ \overbrace{A_{k+1},\ \ldots,\ A_n}^{\text{don't care}})}_{\text{SM}}$$

Figure 4: The design pattern for producing an SM n-ary product.

We introduce one more useful (unary) operator. The algebra FLIP(A) is identical to A in all respects except for its preference relation, which is reversed:

$$\sigma_1 \preceq \sigma_2 \Leftrightarrow \sigma_2 \preceq_A \sigma_1$$

To properly treat the preservation properties of this operator we need to introduce two new monotonicity properties, anti-monotonicity (AM) and strict anti-monotonicity (SAM):

(AM) For all $\sigma \in \Sigma$, for all $\lambda \in L$, $\lambda \oplus \sigma \preceq \sigma$.

(SAM) For all $\sigma \in \Sigma - \{\phi\}$, for all $\lambda \in L$, $\lambda \oplus \sigma \prec \sigma$.

(Note that a routing algebra, such as OP(n), can be both M and AM.) The FLIP(A) operator then has the following preservation property:

A	FLIP(A)
M	AM
AM	M
SM	SAM
SAM	SM

A full treatment of this would involve extending Table 3 to indicate how AM and SAM are preserved. For space reasons we do not do that here. But note that it is fairly easy to show that FLIP($A \otimes B$) is equivalent to FLIP(A) $\otimes$ FLIP(B), so that, at least for product signatures, we can always push this operator down toward the leaf expressions.

Note that the algebra MIN(n) is AM, so that FLIP(MIN(n)) is M. For example, FLIP(MIN(5)) has this $\oplus$ table:

$\oplus$	5	4	3	2	1
1	1	1	1	1	1
2	2	2	2	2	1
3	3	3	3	2	1
4	4	4	3	2	1
5	5	4	3	2	1

Note that the signature 5 is the *most preferred* signature, yet $\oplus$ is defined as $x \oplus y = \min(x,\ y)$. We might define this algebra as

$$\text{WIDTH}(n) \stackrel{\text{def}}{=} \text{FLIP}(\text{MIN}(n)),$$

which would be useful as a monotonic (M) metric for bandwidth. Another useful monotonic algebra is

$$\text{RELIABILITY} \stackrel{\text{def}}{=} \text{FLIP}(\text{MULT}_r(0,\ 1)),$$

which can model the reliablity of a path when reliability estimates are known for each link.

Since monotonicity properties are important, it is useful to have operators that "coerce" any algebra to one in which a monotonicity property holds. The unary operator FM(A) (force monotonicity) eliminates any violations of monotonicity simply by forcing any offenders to take the signature ϕ. Define the $\oplus$ operation as

$$\lambda \oplus \sigma = \begin{cases} \phi & \text{if } \lambda \oplus_A \sigma \prec \sigma \\ \lambda \oplus_A \sigma & \text{otherwise} \end{cases}$$

It should be clear that FM(A) is always monotonic (M). For example, the $\oplus$ table of FM(MIN(5)) is

$\oplus$	1	2	3	4	5	ϕ
1	1	ϕ	ϕ	ϕ	ϕ	ϕ
2	1	2	ϕ	ϕ	ϕ	ϕ
3	1	2	3	ϕ	ϕ	ϕ
4	1	2	3	4	ϕ	ϕ
5	1	2	3	4	5	ϕ

An operator to force SM, FSM(A), is defined in much the same way. The forcing operators do no always produce interesting results. For example, the $\oplus$ table of FSM(MIN(5)) contains only ϕ entries (the algebra MIN(n) is AM). This leads us to consider more gentle techniques for inducing monotonicity.

Suppose we pair up each σ of A with a *level counter* to form tuples of the form $\langle i,\ \sigma \rangle$, which will be compared lexicographically, with the normal integer order $\leq$ used as the preference relation on the first component. The idea is to "bump up" the level counter for any application of $\oplus_A$ that violates a monotonicity constraint. The $\oplus$ operation of LM($n,\ A$) (lift to monotonicity) is defined as

$$\lambda \oplus \langle i,\ \sigma \rangle = \begin{cases} \langle i,\ \lambda \oplus_A \sigma \rangle & \text{if } \sigma \preceq_A \lambda \oplus_A \sigma \\ \langle i+1,\ \lambda \oplus_A \sigma \rangle & \text{if } \lambda \oplus_A \sigma \prec_A \sigma \text{ and } i < n \\ \phi & \text{otherwise} \end{cases}$$

The operator LSM($n,\ A$) (lift to strict monotonicity) is defined in a similar manner. The integer n is simply an upper bound on the level counter. For example, the $\oplus$ table of LM(2, MIN(5)) is presented in Table 4.

4. DOWN WITH BGP!

As mentioned, BGP is currently being used as an IGP. Here we consider how RAML might be used to define more suitable routing protocols. Many applications may need only a RIP-like protocol that scales to large networks and supports complex policy routing. The following routing algebra, MyFirstIGP^A, captures this well.

$$\begin{aligned} \text{PROG}(\otimes(&\textbf{weight} : \text{ADD}(1,\ 2^{32}),\\ &\textbf{router-path} : \text{SIMSEQ}(2^{32},\ 30),\\ &\textbf{tags} : \text{TAGS}(\textit{string}))) \end{aligned}$$

The attribute **router-path** is used both to break ties between routes with equal **weight** and to avoid the problem of *counting to infinity*. This algebra supports programmable labels that can apply policy conditionally:

$$\begin{aligned} \lambda \quad = \quad & \textbf{router-path.label} := 10.10.10.10;\\ & \text{if 'data center' is in } \textbf{tags}\\ & \text{then } \textbf{weight.label} := 20\\ & \text{else if 'sales center' is in } \textbf{tags}\\ & \quad \text{then } \textbf{weight.label} := 30\\ & \quad \text{else reject} \end{aligned}$$

Here the notation **weight.label** denotes the label associated with the **weight** component of the product signature, and "reject" means apply the ϕ label.

Suppose we require an IGP for a very large network that has three levels of (nested) administrative areas. Metropolitan area networks (MANs) link offices within small geographic areas. Regional area networks (RANs) link together MANs within large geographic regions. The Global Network (GN) links together all of the RANs. We would like to hide information (and route flapping) between areas. For this we define the following routing algebra.

$$\begin{aligned} \text{GN}^A \quad &\stackrel{\text{def}}{=} \quad \otimes(\textbf{G-weight} : \text{ADD}(1,\ 2^{32}),\\ &\qquad\quad \textbf{R-path} : \text{SIMSEQ}(2^{32},\ 30),\\ &\qquad\quad \textbf{G-tags} : \text{TAGS}(\textit{string}))\\ \text{RAN}^A \quad &\stackrel{\text{def}}{=} \quad \otimes(\textbf{R-weight} : \text{ADD}(1,\ 2^{32}),\\ &\qquad\quad \textbf{M-path} : \text{SIMSEQ}(2^{32},\ 30),\\ &\qquad\quad \textbf{R-tags} : \text{TAGS}(\textit{string}))\\ \text{MAN}^A \quad &\stackrel{\text{def}}{=} \quad \otimes(\textbf{M-weight} : \text{ADD}(1,\ 2^{32}),\\ &\qquad\quad \textbf{router-id-path} : \text{SIMSEQ}(2^{32},\ 30),\\ &\qquad\quad \textbf{M-tags} : \text{TAGS}(\textit{string}))\\ \text{MyIGP}^A \quad &\stackrel{\text{def}}{=} \quad \text{PROG}(\text{GN}^A \odot (\text{RAN}^A \odot \text{MAN}^A)) \end{aligned}$$

Signatures in the MyIGP^A algebra are 9-tuples that are compared lexicographically. Each signature has the follow-

$\oplus$	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	ϕ
1	(1, 1)	(2, 1)	(2, 1)	(2, 1)	(2, 1)	(2, 1)	ϕ	ϕ	ϕ	ϕ	ϕ
2	(1, 1)	(1, 2)	(2, 2)	(2, 2)	(2, 2)	(2, 1)	(2, 2)	ϕ	ϕ	ϕ	ϕ
3	(1, 1)	(1, 2)	(1, 3)	(2, 3)	(2, 3)	(2, 1)	(2, 2)	(2, 3)	ϕ	ϕ	ϕ
4	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(2, 4)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	ϕ	ϕ
5	(1, 1)	(1, 2)	(1, 3)	(1, 4)	(1, 5)	(2, 1)	(2, 2)	(2, 3)	(2, 4)	(2, 5)	ϕ

Table 4: The $\oplus$ table for LM(2, MIN(5)).

ing form (ignoring attribute names).

$$\overbrace{\langle \overbrace{w_g,\ p_r,\ t_g}^{\text{change between RANs}},\ \langle \overbrace{w_r,\ p_m,\ t_r}^{\text{change between MANs}},\ \overbrace{\langle\ \underbrace{w_m,\ p_i,\ t_m}_{\text{re-initialized between MANs}}\ \rangle}^{\text{change within MANs}} \rangle \rangle}^{\text{re-initialized between RANs}},$$

We have indicated where attributes change and where they are re-initialized. Note that the set of tags associated with routes are independent for each administrative area and "vanish" when routes cross administrative boundaries. If needed, policy can be used to translate tags from one area to another. Routes get originated withing MANs, and a typical origination policy might be to set the regional and global attributes to minimal values.

One could imagine different MANs might require different local algebras. For example, suppose that some MANs want to break ties using bandwidth, others with reliability, while others will use neither of these tie breakers. We might then modify the definition of MAN^A as follows.

$$\begin{aligned}
\text{BW} &\stackrel{\text{def}}{=} \textbf{bandwidth} : \text{WIDTH}(2^{32}) \\
\text{RY} &\stackrel{\text{def}}{=} \textbf{reliability} : \text{RELIABILITY} \\
\text{TIE-BREAK} &\stackrel{\text{def}}{=} \bot(\text{min},\ \text{RY} \lhd \text{BW}) \\
\text{MAN}^A &\stackrel{\text{def}}{=} \otimes(\textbf{M-weight} : \text{ADD}(1,\ 2^{32}), \\
&\qquad \textbf{router-id-path} : \text{SIMSEQ}(2^{32},\ 30), \\
&\qquad \textbf{tie-break} : \text{TIE-BREAK}, \\
&\qquad \textbf{M-tags} : \text{TAGS}(\mathit{string}))
\end{aligned}$$

Note that if different MANs originate different sets of prefixes, then no routes with different tie breaking techniques will be compared. However, if this is not the case then the specification tells us no tie break ($\bot$) will be preferred over reliability, which in turn is preferred over bandwidth.

The IGPs defined above are all SM. This is easy to derive from the monotonicity properties of the base algebras and the preservation properties of the operators used.

5. LONG LIVE BGP!

We now study the policy component of BGP within RAML. Using the scoped product, we define

$$\text{BGP}^A = \text{EBGP}^A \odot \text{IBGP}^A,$$

where EBGP^A models the EBGP policy component, IBGP^A models the IBGP policy component. In this paper we focus only on EBGP and do not go into the details of modeling IBGP.

$$\begin{aligned}
\text{EBGP}^A \stackrel{\text{def}}{=}\ & \text{PROG}(\otimes(\textbf{locpref} : \text{FLIP}(\text{LP}(2^{32})), \\
& \qquad \textbf{aspath} : \text{SIMSEQ}(2^{16},\ 200), \\
& \qquad \textbf{origin} : \text{OP}(3), \\
& \qquad \textbf{med} : \bot(\text{min},\ \text{LP}(2^{32})), \\
& \qquad \textbf{community} : \bot(\text{min},\ \text{TAGS}(\mathit{int}))))
\end{aligned}$$

Figure 5: An RAML expression for the EBGP routing algebra.

The definition of EBGP^A is fairly straightforward, and is presented in Figure 5. The local preference attribute, **locpref**, uses LP(n), but with the preference reversed (larger integers are more preferred). The **aspath** is a simple sequence of AS numbers, where the maximum length is set to 200. The **origin** attribute in BGP has value IGP, EGP, or INCOMPLETE, and the lowest origin type are preferred, where IGP < EGP < INCOMPLETE. We use OP(3) to model this. The **med** attribute (Multi-Exit Discriminator) is used to implement "cold potato" routing. In order to avoid the notorious problems of this attribute [18, 26], we model **med** as if this attribute is always compared, no matter what the next hop ASN.

As it stands, the expression in Figure 5 represents an algebra that is neither M nor SM. But note that if we *replace* the **localpref** component with a expression that is merely monotonic (M), then it follows from the design pattern of Figure 4 that the *entire* expression will be strictly monotonic (SM). This is due to the presence of the SM algebra used for AS paths. Thus, if we can replace the **localpref** component with an M algebra and define an SM algebra for IBGP, then we would have a BGP that is guaranteed to converge, no matter how it is (mis-) configured.

It is known that the standard practice of preferring customer routes over peer routes, and peer routes over provider routes provide at least some protection from BGP divergence [6]. We now demonstrate how easily similar results are obtained using RAML. First, recall that the routing algebra LP(3) has the $\oplus$ table

$\oplus$	1	2	3
1	1	1	1
2	2	2	2
3	3	3	3

For readability, we rename the signatures as follows: 1 to C (customer routes), 2 to R (peer routes), and 3 to P (provider routes). Similarly, we rename the labels as follows: 1 to c (labels a link to a customer), 2 to r (labels a link to a peer), and 3 to p (labels a link to a provider). This produces the $\oplus$ table

$\oplus$	$(1,\ C)$	$(1,\ R)$	$(1,\ P)$	$(2,\ C)$	$(2,\ R)$	$(2,\ P)$	$(3,\ C)$	$(3,\ R)$	$(3,\ P)$	ϕ
c	$(1,\ C)$	$(2,\ C)$	$(2,\ C)$	$(2,\ C)$	$(3,\ C)$	$(3,\ C)$	$(3,\ C)$	ϕ	ϕ	ϕ
r	$(1,\ R)$	$(1,\ R)$	$(2,\ R)$	$(2,\ R)$	$(2,\ R)$	$(3,\ R)$	$(3,\ R)$	$(3,\ R)$	ϕ	ϕ
p	$(1,\ P)$	$(1,\ P)$	$(1,\ P)$	$(2,\ P)$	$(2,\ P)$	$(2,\ P)$	$(3,\ P)$	$(3,\ P)$	$(3,\ P)$	ϕ

Table 5: The $\oplus$ table for $\text{LM}(2,\ \text{LP}(3))$, after $\text{LP}(3)$ signatures have been renamed, $1 \to C$, $2 \to R$, $3 \to P$, and labels renamed, $1 \to c$, $2 \to r$, and $3 \to p$.

$\oplus$	C	R	P
c	C	C	C
r	R	R	R
p	P	P	P

For example, the rule

$$p \oplus C = P$$

can be read as follows: if one of my providers sends me a route from one of its customers, then I will treat it as a provider route. This algebra is not M, but we can transform it to an M algebra by application of the FM operator to obtain

$\oplus$	C	R	P	ϕ
c	C	ϕ	ϕ	ϕ
r	R	R	ϕ	ϕ
p	P	P	P	ϕ

which is very similar to the standard customer/peer/provider rules. For example, $c \oplus R = \phi$ can be read as follows: a customer cannot send one of its peer routes to a provider (or a provider cannot accept a customer's peer routes). In fact, our table is more general than the rules of [6, 25] in that $r \oplus R = R$ and not ϕ. Note that our relationships can be implemented on a *per prefix* basis, since the labels of our algebra are conditional/sequential "programs" over the labels $\{c,\ r, p\}$ (we are using the PROG operator). In other words, using the *generic* operators of RAML, we are able to easily obtain results more general than those of [6, 25] for a model of BGP that captures more detail of the actual protocol. In addition, we see that in the context of the definition of the routing algebra for EBGP (Figure 5), we only need this to be M for the entire expression to be SM — that is, we don't have to assume that there are no customer/provider cycles in the relationship graph as is done [6]. Such cycles, although odd, really pose no problems for convergence.

Instead of forcing monotonicity, let us instead try lifting to monotonicity. Table 5 presents the $\oplus$ table for $\text{LM}(2,\ \text{LP}(3))$, with labels and signatures renamed as above. This is very similar to the scheme presented in [9] to model BGP with backup routes. That model required an entire appendix of tedious correctness proofs. Here we merely apply a generic operator. In addition, our resulting algebra is more general, and includes useful cases that are eliminated in [9]. For example, the rule

$$c \oplus (1,\ P) = (2,\ C)$$

means that a provider can take a route from a customer that the customer is getting from one of its providers (creating a "valley" in the AS path). Of course the level number is increased making the route less preferred. But this would actually represent a potential revenue source for the customer, or a viable emergency routing plan. This type of arrangement is very difficult to implement in BGP today. If this sort of transit is not appropriate in a given AS, then filters can be applied to ensure that this cannot happen (remember, we have the PROG operator providing programmable labels). However, this does seem to provide more flexibility than most implementations today — and this represents potential revenues to ISP.

In fact, there is no reason to force a choice between these options, we could *combine them* into this **locpref** replacement:

$$\textbf{class} : \text{FM}(\text{LP}(3)) \lhd_{t(x)=\langle 1,\ x\rangle} \textbf{lclass} : \text{LM}(k,\ \text{LP}(3))$$

This would allow operators to combine these approaches as they see fit.

Due to space reasons we do not model IBGP in this paper. However, observe that the scoped product operator provides more structure than exists in the current design of BGP. For example, in a RAML model of BGP, all IBGP elements reside in the second component of the scoped product and do not "leak into" the first component. Contrast this with the current BGP design where elements such as the ASPATH for confederations are hacked into the EBGP component, even though they are a part of IBGP. With the scoped product we would naturally define a new attribute for BGP confederation ASPATHs, and the length of this path could then be used in the EBGP route selection process. As with the MyIGP example, a new community set needs to be defined for IBGP, which is not the same as the EBGP community attribute, and this internal community set would automatically vanish at AS boundaries. This actually corresponds closely with common practice among network operators in making a sharp distinction between internal and external communities values.

6. LABEL MODALITIES

So far the labels have been fairly abstract objects associated with arcs between nodes. However, in the world of real protocols it matters how a label gets attached to the network model. This is especially true when two routers that share an adjacency but reside in different administrative domains — this requires some type of cooperation to construct labels.

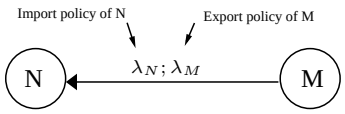

Figure 6: Construction of shared labels.

One way to accomplish this using sequential lables is illustrated in Figure 6. Here we imagine that nodes N and M

are in different administrative domains. We would expect that the label on the link from N to M would be somehow constructed by *both* N and M — or more precisely, by information from the configuration of N *and* from the configuration of M. If λ_N can be constructed from information at N, and λ_M can be constructed from information at M, then $(\lambda_N; \lambda_M) \oplus \sigma = \lambda_N \oplus (\lambda_M \oplus \sigma)$ represents the composition of these policies. That is, λ_N represents an *import policy* at N, and λ_M represents an *export policy* at M.

The details of how this information is presented in configuration associated with N and M, and how these policies are applied are interesting implementation issues, but beyond the scope of the current paper. However, we note that some protocols may require that the export and import rules define a single "atomic" label, not a sequential label. For example, for our model of BGP (Section 5) a sequential label would involve adding *two* ASNs to an ASPATH at each AS boundary, rather than one. It is primarily for this type of situation that we introduce the notion of a *label modality*.

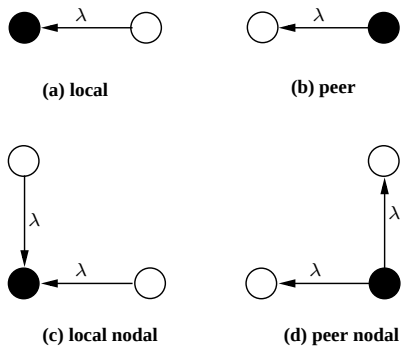

Figure 7: Four label modalities.

Figure 7 illustrates the four major label modalities. In this figure, each filled circle represents the router that controls the value of λ associated with the adjacent arc(s). That is, the configuration of the router represented by the filled circle will produce the associated λ.

Modalities can be associated with individual named attributes of a signature. Using BGP as an example, the **locpref** attribute is an example of a local modality — the receiver of the route attaches the label. On the other hand, **med** is an example of Figure 7(b), called a *peer modality* — the sender of a route attaches the label. The **aspath** attribute is built up from AS numbers that are configured at each node, and as with **med**, the sender attaches the label. However, in this case we will insist that the sender uses the same AS number for all of its BGP sessions, and we call this a *peer nodal modality*. The **origin** attribute provides an example of the *local nodal* modality — the label is local, but the same value must be used in all sessions. Other modalities are also useful. For example, the *constant modality* (always use a fixed label), and the *default modality* (which label to use when one is not supplied by configuration).

7. DISCUSSION AND OPEN PROBLEMS

We are currently implementing a metarouting prototype in the XORP system, which will be described elsewhere. We are implementing two distinct approaches. In the first approach, we simply "hijack" BGP. Additional (optional) attributes are defined for updates, which are sufficiently rich in structure that they can be used to describe routes for any routing algebras defined in RAML. The particular algebra being used by an operator is defined in the router's configuration file and then bound to the appropriate BGP peering sessions. At session initiation, capabilities negotiation includes a check that the same algebra is being used on each end of the session. Of course users are then restricted to using the hard-state, path-vector mechanism of BGP. Our second approach is a bit more ambitious — it will allow users to select not only the routing policy language, but to select mechanisms from a collection of link-state and path-vector implementations and then bind these to routing adjacencies.

The routing algebra metalanguage (RAML) presented in this paper does not represent the only possible choice of base algebras and algebraic operators. Our main design concerns were the ability to express most interesting routing protocols while at the same time retaining the ability to automatically derive monotonicity properties for each algebraic expression in the language. This raises many interesting research questions. For example, is there a *natural* set of base algebras and operators? We would also like to explore RAML operators that preserve the isotonicity properties required of algebras that use a generalized version Dijkstra's algorithm. We suspect that a deeper understanding of the semantics of routing policy metalanguages is required for exploring this and related questions. Intuition suggests that the more expressive an RAML is, the more difficult it will be to automatically determine such global properties (see also [8]).

The RAML should probably include a means of abstraction over algebras having a specified monotonicity property. For example, we defined the BGP routing algebra as

$$\text{BGP}^A = \text{EBGP}^A \otimes \text{IBGP}^A,$$

but we imagine the IBGP component does not actually have to be a part of the global protocol definition. That is, BGP's policy component could be specified as

$$\text{BGP}^A(B : SM) = \text{EBGP}^A \otimes B,$$

where B is a variable ranging over all SM routing algebras — each network could potentially instantiate BGP with a different IBGP.

We have used lexicographic preference on products, but other approaches may prove useful as well. It is interesting to compare the algebra

$$\text{ADD}(1,\ n) \otimes \text{WIDTH}(m),$$

with the (default) policy of EIGRP [23]. In both cases signatures are pairs of the form $\langle d,\ b\rangle$, were d represents distance and b represents bandwidth, and $\oplus$ is defined as

$$\langle d_1,\ b_1\rangle \oplus \langle d_2,\ b_2\rangle = \langle d_1 + d_2,\ min(b_1,\ b_2)\rangle$$

However, our definition uses lexical ordering while EIGRP computes a derived weight, $d+k/b$, for each signature $\langle d,\ b\rangle$ (k is a configured constant). If the first component, d, is strictly greater than zero, then the algebra is SM, and if the first component can be zero then the algebra is only M (see [25] for another discussion of EIGRP). It may be possible to extend RAML with product signatures that do not use lexical preference, but instead base preference on polynomial expressions (as with EIGRP). The difficulty seems

to be finding the right combination of polynomial operators that lend themselves well to an automatic derivation of monotonicity preservation properties.

Routing security is another possible extension to RAML. Security-related extensions to BGP have been defined in SBGP [16]. We feel these are needed extensions, but they are being integrated into monolithic protocol implementations that provide for little or no reuse between protocols. Can generic security-related operators be added to RAML in such a way that certain security properties can be guaranteed for routing?

Algebraic frameworks other than that of [25] may provide foundations for other RAML-like languages. The algebras of [25] have a right-associative $\oplus$ operator, which is well suited for complex policies for path-vector protocols. Perhaps left-associative algebras may prove useful in modeling reservation-based and source-based routing protocols.

We have applied RAML operators at the level of routing algebras, but we believe that operators at the *protocol level* are actually needed to model some protocols and protocol interactions. For example, If $RP_A = \langle A,\ \mathcal{M}_A\rangle$ and $RP_B = \langle B,\ \mathcal{M}_B\rangle$ are two routing protocols (ignoring label modalities). Then the disjunction operator $\lhd$ extends naturally to the protocol level,

$$\langle A,\ \mathcal{M}_A\rangle \lhd_t \langle B,\ \mathcal{M}_B\rangle = \langle A \lhd_t B,\ \mathcal{M}_A ? \mathcal{M}_B\rangle.$$

The ? here suggests that some means for combining mechanisms in a consistent way needs to be worked out. Put another way, we can think of this combination as defining a *single protocol*. To treat this in full detail seems to require that *protocol instances*, the FIB, and the FIB manager be brought into our formal model. Then we would think of the preference relation at the FIB-manager level as modeling what is usually called *administrative distance* between routing protocols. In fact, we believe this is the most promising approach to modeling a protocol such as OSPF. We would like to declare OSPF with a specification something like this,

$$\text{OSPF} = \langle \textsc{areas},\ \text{path-vector}\rangle \odot \langle \textsc{add}(1,\ 2^{32}),\ \text{link-state}\rangle,$$

where the algebra AREAS could be defined as $\textsc{fm}(\textsc{lp}(2)) \otimes \textsc{add}(1,\ 3)$ (the first component enforces a simple hierarchy, while the second restricts path lengths to length at most 3). As mentioned in Section 3, an OSPF signature can be modeled as a pair, $\langle \alpha, d\rangle$, where α contains area information and d represents distance.

Acknowledgements

Initial work for this paper was done while the first author was with Intel Research, and we thank Derek McAuley and David Tennenhouse for their support. We also thank Andrea Bittau, Steve Bellovin, Bob Briscoe, Randy Bush, Chikin (Sid) Chau, Jon Crowcroft, Nick Feamster, Wenjun Hu, Richard Mortier, Kengo Nagahashi, Matthew Roughan, Peter Sewell, Bruce Shepherd, Ben Strulo, Nigel Walker, and Gordon Wilfong for helpful comments on early drafts.

8. REFERENCES

[1] R. Callon. Use of OSI IS–IS for Routing in TCP/IP and Dual Environments. RFC1195, December 1990.

[2] Bernard Carré. *Graphs and Networks*. Oxford University Press, 1979.

[3] Dan Decasper, Zubin Dittia, Guru Parulkar, and Bernhard Plattner. Router plugins: a software architecture for next generation routers. *SIGCOMM Comput. Commun. Rev.*, 28(4):229–240, 1998.

[4] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik 1*, pages 269–271, 1959.

[5] Forces. IETF Forwarding and Control Element Separation working group.

[6] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Transactions on Networking*, pages 681–692, December 2001.

[7] M. Gondran and M. Minoux. *Graphs and Algorithms*. Wiley, 1984.

[8] Timothy Griffin, Aaron D. Jaggard, and Vijay Ramachandran. Design principles of policy languages for path vector protocols. In *Proc. ACM SIGCOMM*, September 2003.

[9] Timothy G. Griffin, Lixin Gao, and Jennifer Rexford. Inherently safe backup routing with BGP. In *Proc. IEEE INFOCOM*, April 2001.

[10] Timothy G. Griffin and Geoff Huston. BGP Wedgies, June 2005. Internet Draft (work in progress).

[11] Sam Halabi and Danny McPherson. *Internet Routing Architectures*. Cisco Press, second edition, 2001.

[12] Mark Handley, Eddie Kohler, Atanu Ghosh, Orion Hodson, and Pavlin Radoslavov. Designing extensible IP router software. In *2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2005. see also `www.xorp.org`.

[13] C. Hendrick. Routing information protocol. RFC 1058, 1988.

[14] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000.

[15] K.Varadhan, R.Govindan, and D Estrin. Persistent route oscillations in inter-domain routing. *Computer Networks*, 32:1–16, 2000.

[16] Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure BGP (S-BGP). draft-clynn-s-bgp-protocol-00a.txt. work in progress.

[17] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[18] D. McPherson, V. Gill, D. Walton, and A. Retana. RFC3345: BGP persistent route oscillation condition, 2002.

[19] Mehryar Mohri. Semiring frameworks and algorithms for shortest-distance problems. *J. Autom. Lang. Comb.*, 7(3):321–350, 2002.

[20] J. Moy. OSPF version 2. RFC 2328, 1998.

[21] John Moy. *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley, 1998.

[22] Craig Partridge, Alex C. Snoeren, W. Timothy Strayer, Beverly Schwartz, Matthew Condell, and Isidro Castineyra. Fire: Flexible intra-as routing environment. *IEEE Journal on Selected Areas in Communications (J-SAC)*, 19(3), 2001. A preliminary version appeared in SIGCOMM 2000.

[23] Albaro Retana, Russ White, and Don Slice. *EIGRP for IP*. Addison-Wesley, 1998.

[24] Joao Luis Sobrinho. Algebra and algorithms for QoS path computation and hop-by-hop. *IEEE/ACM Transactions on Networking*, 10(4):541–550, August 2002.

[25] Joao Luis Sobrinho. Network routing with path vector protocols: Theory and applications. In *Proc. ACM SIGCOMM*, September 2003.

[26] Cisco Systems. Endless BGP convergence problem in Cisco IOS software releases. Field Note, October 10 2001, `http://www.cisco.com/warp/public/770/ fn12942.html`.

[27] Russ White, Danny McPherson, and Srihari Sangli. *Practical BGP*. Addison Wesley, 2005.

[28] Randy Zhang and Micah Bartell. *BGP Design and Implementation*. Cisco Press, 2003.