

Metarouting

Timothy G. Griffin
Computer Laboratory
University of Cambridge
Cambridge, UK

João Luís Sobrinho
Instituto de Telecomunicações
Instituto Superior Técnico
Lisbon, Portugal

SIGCOMM 2005

Presented by
Zheng Zeng CS 591 SN

What is Metarouting?

- A new approach to the definition and deployment of routing protocols.
- Abstraction
 - Capture (almost) all routing protocol instances

Why do this?

- No one-size-fits-all routing protocol
- Hard to define, standardize, and deploy new routing protocols (or minor modifications to existing protocols)
 - Just leave it up to operator community to standardize protocols using high-level specs...
- It's fun!

Three Ideas

#1 Decompose routing protocol
into separate parts

Idea #1

Protocol = Mechanism + Policy + ??? ...

- How are routing messages exchanged and propagated?
- What type of route selection algorithms are used?
- ...

- How are the attributes of a route described?
- How are best routes selected?
- ...



RP = < A, M, ??? ... >

Policy

Mechanism

Three Ideas

- #1 Decompose routing protocol into separate parts
- #2 Use "Routing Algebras" as basis for Policy Component

Routing Algebras

How paths are transformed by application of labels

L : link labels

$$\oplus : L \times \Sigma \rightarrow \Sigma$$

$$A = \left(\underbrace{\Sigma, \leq}_{\text{red}}, \underbrace{L, \oplus}_{\text{black}}, \underbrace{0}_{\text{blue}} \right)$$

How paths are described and compared

Σ : path signatures

\leq is a preference relation over Σ :

(complete) $\forall x, y$ in Σ , $x \leq y$ or $y \leq x$ (or both)

(transitive) $\forall x, y, z$ in Σ , if $x \leq y$ and $y \leq z$
then $x \leq z$

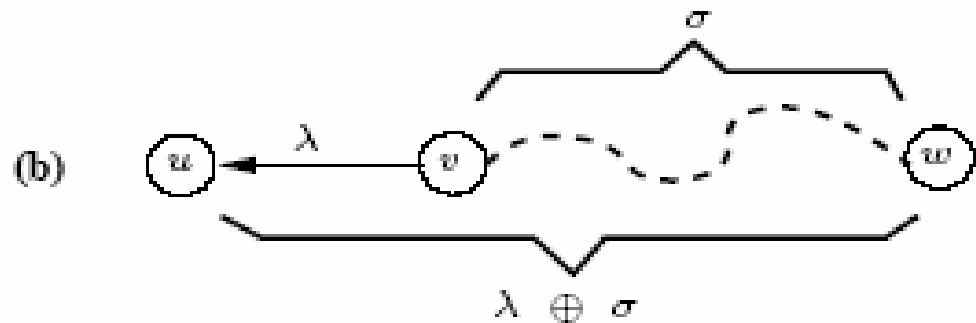
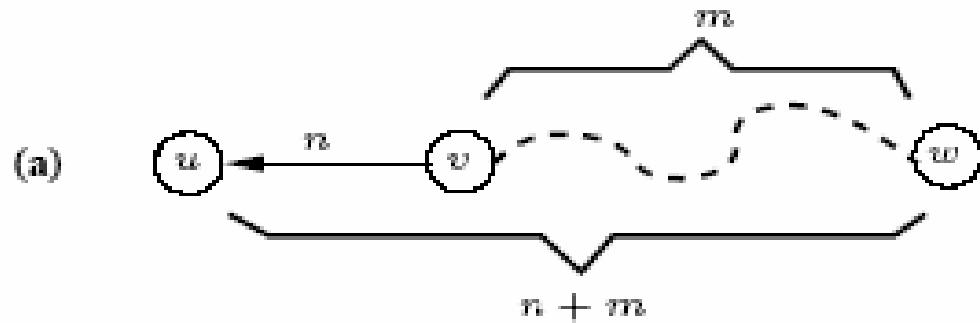
A subset of signatures that can be associated with originated routes.

Shortest path routing

$$A = (\Sigma, \leq, L, +, 0)$$

$$\lambda \in L, \sigma \in \Sigma$$

Generalize
Shortest Paths

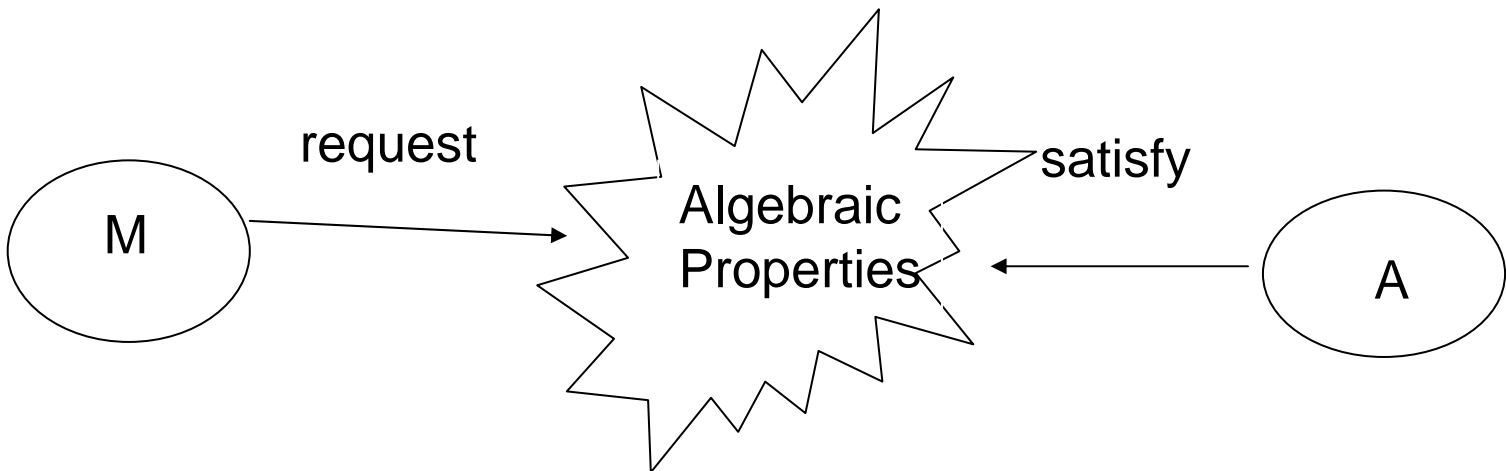


Correctness

We want protocols that are correct!

always converge

RP = < A, M, ??? ... >



Correctness

Monotonicity (M): $\forall \sigma \in \Sigma/\phi, \lambda \in L \quad \sigma \leq \lambda \oplus \sigma$

Strict Monotonicity (SM): $\forall \sigma \in \Sigma/\phi, \lambda \in L \quad \sigma < \lambda \oplus \sigma$

Isotonicity (I): $\forall \sigma, \beta \in \Sigma/\phi, \lambda \in L \quad \sigma \leq \beta \rightarrow \lambda \oplus \sigma \leq \lambda \oplus \beta$

| | SM | I | Assoc. \oplus |
|---|----|---|-----------------|
| vectoring | ★ | | |
| Link-state with generalized Dijkstra | ★ | ★ | ★ |
| Link-state with local vector simulation | ★ | | |

Routing Algebras are a good start, but...

- The algebraic framework does not, by itself, provide a way of constructing new and complex algebras.
 - Algebra definition is hard...
 - Proofs are tedious...
 - Modifications to an algebra's definitions are difficult to manage...

Three Ideas

- #1 Decompose routing protocol into separate parts
- #2 Use "Routing Algebras" as basis for Policy Component
- #3 Use RAML to define new and more complex algebras from simple ones.

Idea #3

Routing Algebra Meta-Language (RAML)

| | |
|---------------------|--------------------|
| $A ::= B$ | (base algebras) |
| $ \text{Op}(A)$ | (unary operator) |
| $ A \text{ Op } A$ | (binary operators) |

- Goals

- Want to **automatically** derive properties (M, SM, ...) of the algebra represented by an RAML expression from properties of base algebras and **preservation properties** of operators

Example: $A1 = *(B1)$

- Simplicity
- Expressiveness

Base Algebra Example Addition (ADD)

$$\text{ADD}(n, m) : (S, \leq, L, \oplus, 0)$$

$$L = \mathcal{O} = \{n, \dots, m\} \text{ and } \Sigma = \{n, \dots, m\} \cup \{\phi\}$$

ϕ the least preferred

ADD(1, 3)

| | | Σ | | | | |
|-----|---|----------|--------|--------|--------|--------|
| | | \oplus | 1 | 2 | 3 | ϕ |
| L { | 1 | | 2 | 3 | ϕ | ϕ |
| | 2 | | 3 | ϕ | ϕ | ϕ |
| | 3 | | ϕ | ϕ | ϕ | ϕ |

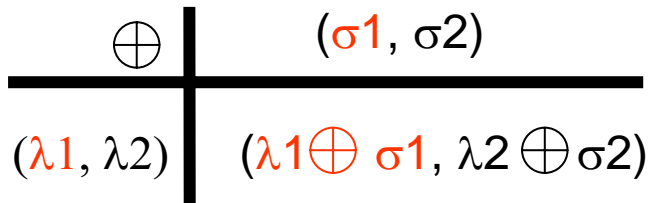
ADD(n, m) is SM if $0 < n \leq m$

Other Base Algebras

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

OP Example: Lexical Product

$A \otimes B$



$$(\phi, _) = (_, \phi) = \phi$$

Preference is Lexical order

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

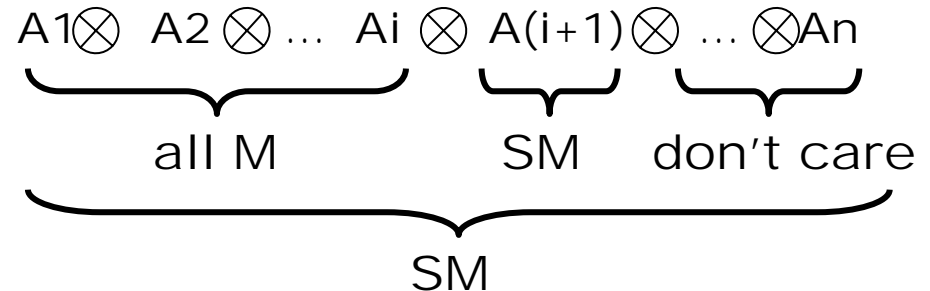
$$\begin{array}{c} \Sigma \\ ((\Sigma_A - \{\phi\}) \times (\Sigma_B - \{\phi\})) \cup \{\phi\} \\ L \\ L_A \times L_B \end{array}$$

Preservation properties

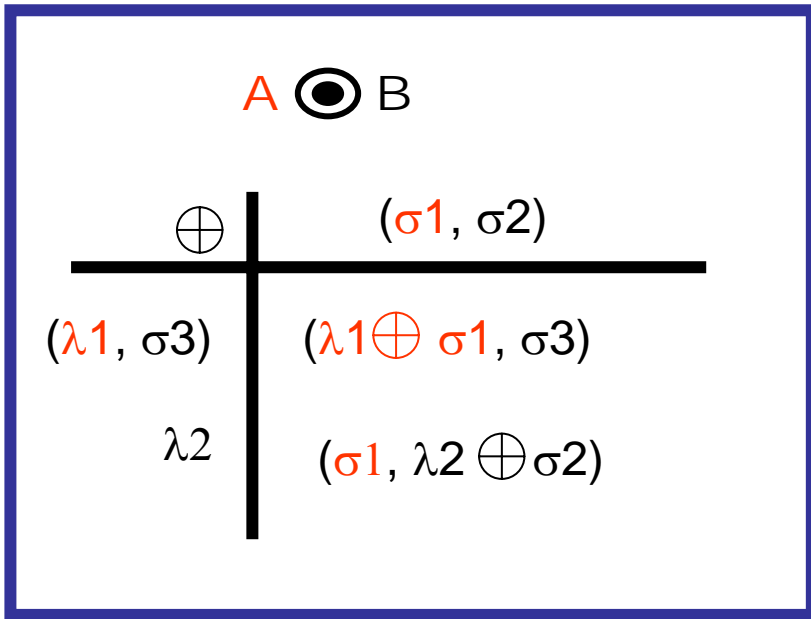
| A | B | $A \otimes B$ |
|-----|-----|---------------|
| M | M | M |
| SM | | SM |
| M | SM | SM |



This suggests a design pattern for SM:



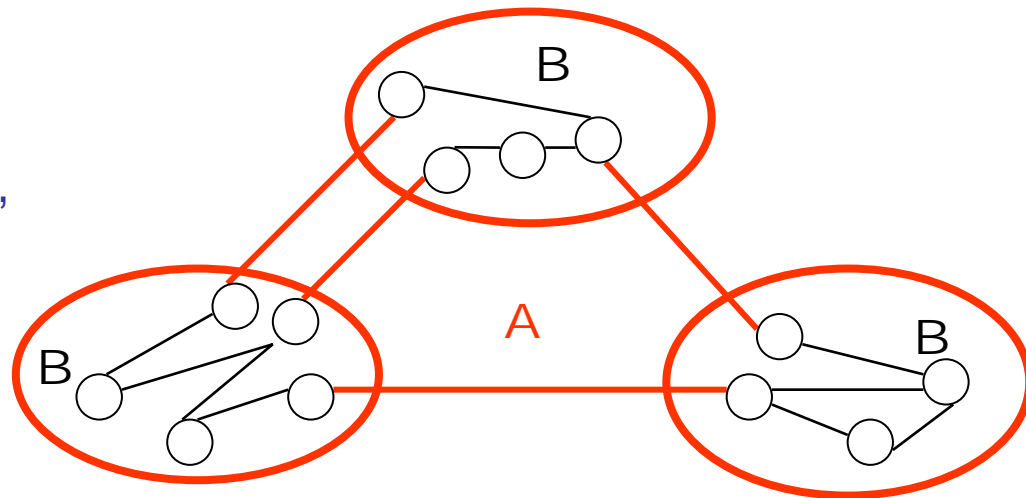
Scoped Product



Preservation properties

| A | B | $A \odot B$ |
|-----|-----|-------------|
| SM | SM | SM |
| SM | M | M |

Can be used to implement IBGP/EBGP-like “information hiding”



Programmatic Labels

$$A = (\Sigma, \mathbf{8}, L, \oplus, \theta) \quad \text{prog}(A) = (\Sigma, \mathbf{8}, L, \oplus, \theta)$$

$$\lambda ::= \lambda \mid \lambda_1; \lambda_2 \mid \text{reject} \mid \text{if } \pi \text{ then } \lambda_1 \text{ else } \lambda_2$$

$$\lambda \oplus \sigma = \lambda \oplus \sigma$$

$$(\lambda_1; \lambda_2) \oplus \sigma = \lambda_1 \oplus (\lambda_2 \oplus \sigma)$$

$$\text{reject} \oplus \sigma = \phi$$

$$(\text{if } \pi \text{ then } \lambda_1 \text{ else } \lambda_2) \oplus \sigma$$

$$= \begin{cases} \lambda_1 \oplus \sigma & \text{if } \pi(\sigma) \\ \lambda_2 \oplus \sigma & \text{o.w.} \end{cases}$$

| A | prog(A) |
|----|---------|
| M | M |
| SM | SM |

MyFirstIGP

```
PROG( $\otimes$ (weight : ADD(1, 232),  
router-path : SIMSEQ(232, 30),  
tags : TAGS(string)))
```

```
if 'data center' is in tags  
then weight.label := 20  
else if 'sales center' is in tags  
then weight.label := 30  
else reject
```

This is SM

Ongoing, Future Work

- RAML
 - More operators...

- At the "protocol level"

$$\langle A, M1 \rangle \blacktriangleleft \langle B, M2, \rangle = \langle A \blacktriangleleft B, M1 + M2 \rangle$$

↑
Not sure what
"+" means

Ongoing, Future Work

- Implementation
 - Using XORP (www.xorp.org)
 - With Mark Handley (UCL) and others