

Applied Metarouting

John Billings
(with Tim Griffin, Alex Gurney, Sam Hym and Peter Sewell)

University of Cambridge

PRESTO

Problem...

Existing routing protocols could be improved.
(We're talking about: RIP, OSPF, IS-IS, EIGRP and BGP)

Existing routing protocols could be improved.
(We're talking about: RIP, OSPF, IS-IS, EIGRP and BGP)

- ▶ **Broken**

Existing routing protocols could be improved.
(We're talking about: RIP, OSPF, IS-IS, EIGRP and BGP)

- ▶ **Broken**
 - ▶ BGP has no guarantee of convergence

Existing routing protocols could be improved.
(We're talking about: RIP, OSPF, IS-IS, EIGRP and BGP)

- ▶ **Broken**

- ▶ BGP has no guarantee of convergence
- ▶ This is observed in practice

Existing routing protocols could be improved.
(We're talking about: RIP, OSPF, IS-IS, EIGRP and BGP)

- ▶ **Broken**
 - ▶ BGP has no guarantee of convergence
 - ▶ This is observed in practice
- ▶ **Inflexible**

Existing routing protocols could be improved.
(We're talking about: RIP, OSPF, IS-IS, EIGRP and BGP)

- ▶ **Broken**

- ▶ BGP has no guarantee of convergence
- ▶ This is observed in practice

- ▶ **Inflexible**

- ▶ Operators have resorted to deploying BGP as an IGP

Existing routing protocols could be improved.
(We're talking about: RIP, OSPF, IS-IS, EIGRP and BGP)

- ▶ **Broken**

- ▶ BGP has no guarantee of convergence
- ▶ This is observed in practice

- ▶ **Inflexible**

- ▶ Operators have resorted to deploying BGP as an IGP
- ▶ (No customer/peer/provider preference safety-net)

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights
 - ▶ Function for concatenation

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights
 - ▶ Function for concatenation
 - ▶ Function for comparison

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights
 - ▶ Function for concatenation
 - ▶ Function for comparison
 - ▶ **Algorithm** to solve *routing problem*

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights
 - ▶ Function for concatenation
 - ▶ Function for comparison
 - ▶ **Algorithm** to solve *routing problem*
 - ▶ Link state (OSPF, IS-IS)

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights
 - ▶ Function for concatenation
 - ▶ Function for comparison
 - ▶ **Algorithm** to solve *routing problem*
 - ▶ Link state (OSPF, IS-IS)
 - ▶ Distance Vector (RIP, EIGRP, BGP)

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights
 - ▶ Function for concatenation
 - ▶ Function for comparison
 - ▶ **Algorithm** to solve *routing problem*
 - ▶ Link state (OSPF, IS-IS)
 - ▶ Distance Vector (RIP, EIGRP, BGP)
- ▶ Idea

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights
 - ▶ Function for concatenation
 - ▶ Function for comparison
 - ▶ **Algorithm** to solve *routing problem*
 - ▶ Link state (OSPF, IS-IS)
 - ▶ Distance Vector (RIP, EIGRP, BGP)
- ▶ Idea
 - ▶ Create a language to specify the algebra and algorithm

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights
 - ▶ Function for concatenation
 - ▶ Function for comparison
 - ▶ **Algorithm** to solve *routing problem*
 - ▶ Link state (OSPF, IS-IS)
 - ▶ Distance Vector (RIP, EIGRP, BGP)
- ▶ Idea
 - ▶ Create a language to specify the algebra and algorithm
 - ▶ Use machine-checkable conditions to guarantee convergence

Metarouting! (Griffin and Sobrinho, SIGCOMM 2005)

- ▶ What is a routing protocol?
 - ▶ **Algebra**
 - ▶ Set of link weights
 - ▶ Function for concatenation
 - ▶ Function for comparison
 - ▶ **Algorithm** to solve *routing problem*
 - ▶ Link state (OSPF, IS-IS)
 - ▶ Distance Vector (RIP, EIGRP, BGP)
- ▶ Idea
 - ▶ Create a language to specify the algebra and algorithm
 - ▶ Use machine-checkable conditions to guarantee convergence
 - ▶ Compile to a fast implementation

Examples

▶ MR-RIP:

```
let sp : semiring = intMinPlus (0, 15)
```

- ▶ MR-RIP:

```
let sp : semiring = intMinPlus(0, 15)
```

- ▶ MR-RIP with tie-break on bandwidth:

```
let bw : semiring = intMaxMin(0, 100)
```

```
let spLexBw : semiring = omegaLexProduct(sp, bw)
```

- ▶ MR-RIP:

```
let sp : semiring = intMinPlus(0, 15)
```

- ▶ MR-RIP with tie-break on bandwidth:

```
let bw : semiring = intMaxMin(0, 100)  
let spLexBw : semiring = omegaLexProduct(sp, bw)
```

- ▶ MR-BGP: (doesn't fit here ;-)

Grammar generating some of our algebraic expressions:

Grammar generating some of our algebraic expressions:

<i>expr</i>	::=	orAnd	reachability
		intMinPlus(<i>m</i> , <i>n</i>)	shortest-path
		intMaxMin(<i>m</i> , <i>n</i>)	bandwidth
		omegaLexProduct(<i>expr</i> , <i>expr'</i>)	
		setIntersectUnion(<i>n</i> , <i>type</i>)	
		...	

Grammar generating some of our algebraic expressions:

<i>expr</i>	::=	orAnd	reachability
		intMinPlus(<i>m</i> , <i>n</i>)	shortest-path
		intMaxMin(<i>m</i> , <i>n</i>)	bandwidth
		omegaLexProduct(<i>expr</i> , <i>expr'</i>)	
		setIntersectUnion(<i>n</i> , <i>type</i>)	
		...	

- ▶ Countably-infinite set of algebras

Grammar generating some of our algebraic expressions:

<i>expr</i>	::=	orAnd	reachability
		intMinPlus(<i>m</i> , <i>n</i>)	shortest-path
		intMaxMin(<i>m</i> , <i>n</i>)	bandwidth
		omegaLexProduct(<i>expr</i> , <i>expr'</i>)	
		setIntersectUnion(<i>n</i> , <i>type</i>)	
		...	

- ▶ Countably-infinite set of algebras
- ▶ Some of the algebras lead to protocols that do not converge...

Grammar generating some of our algebraic expressions:

<i>expr</i>	::=	orAnd	reachability
		intMinPlus(<i>m</i> , <i>n</i>)	shortest-path
		intMaxMin(<i>m</i> , <i>n</i>)	bandwidth
		omegaLexProduct(<i>expr</i> , <i>expr'</i>)	
		setIntersectUnion(<i>n</i> , <i>type</i>)	
		...	

- ▶ Countably-infinite set of algebras
- ▶ Some of the algebras lead to protocols that do not converge...
- ▶ ... so we use a 'type system' to throw them out!

Grammar generating some of our algebraic expressions:

<code>expr ::=</code>	<code>orAnd</code>	reachability
	<code>intMinPlus(m, n)</code>	shortest-path
	<code>intMaxMin(m, n)</code>	bandwidth
	<code>omegaLexProduct(expr, expr')</code>	
	<code>setIntersectUnion(n, type)</code>	
	<code>...</code>	

- ▶ Countably-infinite set of algebras
- ▶ Some of the algebras lead to protocols that do not converge...
- ▶ ... so we use a 'type system' to throw them out!
- ▶ *Automatically* verify convergence by examining syntax

Grammar generating some of our algebraic expressions:

<code>expr ::=</code>	<code>orAnd</code>	reachability
	<code>intMinPlus(m, n)</code>	shortest-path
	<code>intMaxMin(m, n)</code>	bandwidth
	<code>omegaLexProduct(expr, expr')</code>	
	<code>setIntersectUnion(n, type)</code>	
	<code>...</code>	

- ▶ Countably-infinite set of algebras
- ▶ Some of the algebras lead to protocols that do not converge...
- ▶ ... so we use a 'type system' to throw them out!
- ▶ *Automatically* verify convergence by examining syntax
- ▶ (*not* the global configuration of our network e.g. BGP)

Are you for real?

- ▶ Develop the mathematics

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress
- ▶ Interfaces: Take a routing protocol and slice out everything that deals with link weights. Use as canonical LS / DV algorithm.

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress
- ▶ Interfaces: Take a routing protocol and slice out everything that deals with link weights. Use as canonical LS / DV algorithm.
 - ▶ Done for XORP RIP using SIGCOMM 2005 algebras

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress
- ▶ Interfaces: Take a routing protocol and slice out everything that deals with link weights. Use as canonical LS / DV algorithm.
 - ▶ Done for XORP RIP using SIGCOMM 2005 algebras
 - ▶ To do for OSPF

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress
- ▶ Interfaces: Take a routing protocol and slice out everything that deals with link weights. Use as canonical LS / DV algorithm.
 - ▶ Done for XORP RIP using SIGCOMM 2005 algebras
 - ▶ To do for OSPF
- ▶ Algebras: Write a compiler from our algebraic expressions to efficient C code. Slot into sliced routing protocol.

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress
- ▶ Interfaces: Take a routing protocol and slice out everything that deals with link weights. Use as canonical LS / DV algorithm.
 - ▶ Done for XORP RIP using SIGCOMM 2005 algebras
 - ▶ To do for OSPF
- ▶ Algebras: Write a compiler from our algebraic expressions to efficient C code. Slot into sliced routing protocol.
 - ▶ Work in progress

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress
- ▶ Interfaces: Take a routing protocol and slice out everything that deals with link weights. Use as canonical LS / DV algorithm.
 - ▶ Done for XORP RIP using SIGCOMM 2005 algebras
 - ▶ To do for OSPF
- ▶ Algebras: Write a compiler from our algebraic expressions to efficient C code. Slot into sliced routing protocol.
 - ▶ Work in progress
- ▶ Write some wacky routing protocols

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress
- ▶ Interfaces: Take a routing protocol and slice out everything that deals with link weights. Use as canonical LS / DV algorithm.
 - ▶ Done for XORP RIP using SIGCOMM 2005 algebras
 - ▶ To do for OSPF
- ▶ Algebras: Write a compiler from our algebraic expressions to efficient C code. Slot into sliced routing protocol.
 - ▶ Work in progress
- ▶ Write some wacky routing protocols
 - ▶ Work in progress

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress
- ▶ Interfaces: Take a routing protocol and slice out everything that deals with link weights. Use as canonical LS / DV algorithm.
 - ▶ Done for XORP RIP using SIGCOMM 2005 algebras
 - ▶ To do for OSPF
- ▶ Algebras: Write a compiler from our algebraic expressions to efficient C code. Slot into sliced routing protocol.
 - ▶ Work in progress
- ▶ Write some wacky routing protocols
 - ▶ Work in progress
 - ▶ We'd be interested in hearing from you!

Are you for real?

- ▶ Develop the mathematics
 - ▶ Work in progress
- ▶ Interfaces: Take a routing protocol and slice out everything that deals with link weights. Use as canonical LS / DV algorithm.
 - ▶ Done for XORP RIP using SIGCOMM 2005 algebras
 - ▶ To do for OSPF
- ▶ Algebras: Write a compiler from our algebraic expressions to efficient C code. Slot into sliced routing protocol.
 - ▶ Work in progress
- ▶ Write some wacky routing protocols
 - ▶ Work in progress
 - ▶ We'd be interested in hearing from you!

The end