Looking for Good Abstractions

timothy.griffin@cl.cam.ac.uk

INFOCOM 2006 Infrastructure Routing Panel
April 2006
What Problem is BGP Solving?

<table>
<thead>
<tr>
<th>Underlying problem</th>
<th>Distributed means of computing a solution.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shortest Paths</td>
<td>RIP, OSPF, IS-IS</td>
</tr>
<tr>
<td>Stable Paths</td>
<td>BGP</td>
</tr>
</tbody>
</table>

[GSW1998, GSW2002]
An instance of the *Stable Paths Problem*

most preferred

...  

least preferred (not null)
A Solution to a Stable Paths Problem
If an instance of SPP has an acyclic dispute digraph, then

- Static (SPP):
  - solvable
  - unique solution
  - all sub-problems uniquely solvable

- Dynamic (path-vector):
  - safe (can’t diverge)
  - predictable
  - restoration robust with respect to link/node failures
Disaster strikes primary link and the backup takes over.

Primary link is restored but some traffic remains *pinned* to backup.
Routing Algebras
João Luís Sobrinho

\[ A = (\Sigma, \oplus, \otimes, 0, 1) \]

\[ A = (\Sigma, \leq, \otimes) \]

\[ A = (\Sigma, \leq, \Lambda, \otimes) \]

Path Algebras ---
1970’s, 1980s
Gondran, Minoux, Carre’, ...


2005: An Algebraic Theory of Dynamic Network Routing (TON)
Metarouting
(Griffin, Sobrinho SIGCOMM 2005)

Can we do for routing protocols what YACC did for parsers?

Grammar Specification → YACC → Parser Implementation

protocol Specification → Metarouting Magic → Protocol Implementation
let prefix : algebra =
  op(isolate(IPv4))

let lp3 : algebra =
  lp(min(0,3))

let cpp : algebra =
  fm(lp3)

let node_path : algebra =
  slists(100, strings(20))

let community_set : algebra =
  tags(100, 20)

let sp : algebra =
  add(1, 1000)

let ebgp : algebra =
  lex <
    nlri : prefix,
    loc : cpp,
    path : node_path,
    comm : community_set,
    d : lp(sp),
    ipath : lp(node_path),
    icomm : lp(community_set)
  >

let ibgp : algebra =
  lex <
    nlri : prefix,
    loc : op(cpp),
    path : op(node_path),
    comm : op(community_set),
    d : sp,
    ipath : node_path,
    icomm : community_set
  >

let bgp : algebra = lunion <ebgp : ebgp, ibgp : ibgp>
Familiar things through fresh eyes

**Default Administrative Distance**

<table>
<thead>
<tr>
<th>Route Type</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>direct interface</td>
<td>0</td>
</tr>
<tr>
<td>static route</td>
<td>1</td>
</tr>
<tr>
<td>EIGRP Summary Route</td>
<td>5</td>
</tr>
<tr>
<td>External BGP</td>
<td>20</td>
</tr>
<tr>
<td>IGRP</td>
<td>100</td>
</tr>
<tr>
<td>OSPF</td>
<td>110</td>
</tr>
<tr>
<td>IS-IS</td>
<td>115</td>
</tr>
<tr>
<td>RIP</td>
<td>120</td>
</tr>
<tr>
<td>I-BGP</td>
<td>200</td>
</tr>
</tbody>
</table>

\[ A \equiv \prod_{i \in \{0,1,\ldots,n\}} A_i \]
Open Problems

• Good metarouting Language design
• Addressing and Forwarding?
• Tunnels as first-class objects
• 2547-ish VPNs?