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
A sufficient condition for sanity

If an instance of SPP has an acyclic dispute digraph, then

<table>
<thead>
<tr>
<th>Static (SPP)</th>
<th>Dynamic (path-vector)</th>
</tr>
</thead>
<tbody>
<tr>
<td>solvable</td>
<td>safe (can’t diverge)</td>
</tr>
<tr>
<td>unique solution</td>
<td>predictable</td>
</tr>
<tr>
<td>all sub-problems uniquely solvable</td>
<td>restoration</td>
</tr>
<tr>
<td></td>
<td>robust with respect to link/node failures</td>
</tr>
</tbody>
</table>
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

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

A = (Σ, ⊕, ⊗, 0, 1)


A = (Σ, ≤, ⊗)


A = (Σ, ≤, Λ, ⊗)

2005: An Algebraic Theory of Dynamic Network Routing (TON)
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>
**Default Administrative Distance**

<table>
<thead>
<tr>
<th>Direct Interface</th>
<th>0</th>
</tr>
</thead>
<tbody>
<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?