Design Principles of Policy Languages for Path-Vector Protocols

Timothy G. Griffin (Intel, formerly AT&T Research),
Aaron D. Jaggard (Tulane, formerly UPenn), and
Vijay Ramachandran (Yale)

Partially supported by the CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR grant N00014-01-1-0795.

Some of these results have previously appeared in *Proc. SIGCOMM '03*.

Overview

- BGP used for interdomain routing
  - Interaction of locally configured policies can cause global routing anomalies
  - No design framework
- Develop design principles for protocols and their policy-configuration languages
  - Identify and formally define desirable properties
  - Determine which protocol components and constraints influence those properties
  - Show inherent trade-offs in design space

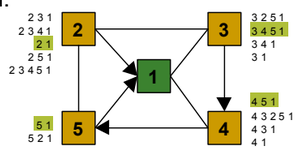
Relevance to Critical Infrastructure Protection

- The Internet or Internet-like networks running path-vector routing protocols are often used to operate our critical infrastructure.
- To secure these networks against malicious or accidental local-configuration changes that affect the rest of the network, we must understand what exactly can cause global routing anomalies and how to prevent them.

Expressiveness

- **Desire:** Maximal expressiveness without sacrificing other design goals
- **Measure:** How many routing configurations can be expressed? Use SPP [GSW02]* as a semantic domain.

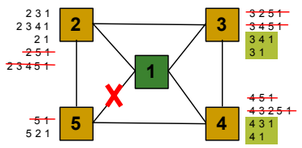
* Griffin, Shepherd, and Wilfong. "The Stable Paths Problem and Interdomain Routing." *IEEE/ACM TON* **10(2)**, 2002.



Robustness

- **Definition:** Unique solvability for any configuration, even after link and node failures
- Primary constraint on expressiveness

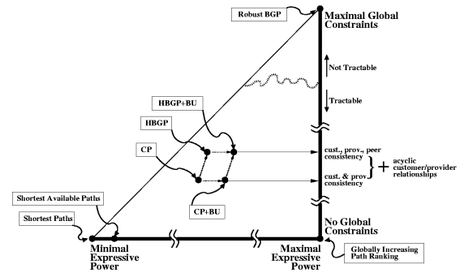
Deleting edge {1, 5} results in a configuration with no predictable solution.



Other Design-Space Dimensions

- **Transparency:** Can policy writers understand the effects of their policies?
- **Policy Opaqueness:** Can routing-policy details remain private?
- **Autonomy:** How much independence do policy writers have in configuration?
- **Global Constraint:** What global conditions must be checked?

A Slice of the Design Space: Robust and Transparent Systems



Path-Vector Policy Systems

Formal model of path-vector routing:

(PV, PL, K)

Path-Vector System:
The underlying message-exchange system for route information. What is exchanged and how?

Global Constraint:
What assumptions about the network must be true to achieve robustness?

Policy Language:
How can policies be described? *PL* acts as a *local constraint* on the expressiveness of policies.

Question:
What role do these components play in achieving our design goals?

Achieving Robustness

- Are there local constraints that guarantee robustness?
  - Conjecture: No PVPS can capture all robust configurations.
  - Theorem: Systems in which extending paths increases their absolute rank are robust.
- The increasing condition precludes achieving other design goals. How do we implement it?
  - Share information and check rank locally: lose autonomy and policy opaqueness
  - Let protocol filter non-increasing routes: lose transparency
  - Have a mechanism to collect all policies and globally check the increasing condition: generally intractable.

Summary of Trade-offs

- **Theorem:** A path-vector policy system (without global constraint) expressive enough to capture all increasing configurations either *does not support autonomy of neighbor ranking* or is *not transparent*, or both.
- **Theorem:** A transparent, robust path-vector policy system that supports autonomy of neighbor ranking and is at least as expressive as shortest paths *must have a non-trivial global constraint*.

HBGP and Class Based PVPSs

- Hierarchical BGP [Griffin, *et al.*, using SPP]
  - Classify neighbor as customer, peer, or provider
  - Avoid customer-provider cycles (implicitly a global constraint; naturally enforced by economics)
- Generalize this in PVPS context
  - Classify neighbors
    - Ranking and exporting based on these classes
  - Employ some sort of global constraint
  - Theorem: Using a template (PV, PL), class-based PVPSs are robust if certain cycles do not exist.

Conclusions

- Formalization of the protocol design space
  Identification of design goals and definitions of protocol components, policy languages and global constraints
- Trade-offs between design parameters
  Local constraints alone not enough to achieve robustness and other design goals concurrently
- Class-based systems
  Sufficient condition for robustness using a general scheme modeling Internet business relationships, but this condition may be hard to check