

Design Principles of Policy Languages for Path-Vector Protocols

Timothy G. Griffin¹ (Intel Research, Cambridge), Aaron D. Jagard^{1,2} (Tulane), and Vijay Ramachandran^{1,2} (Yale) ¹SIGCOMM 2003 and ²CNP 2004
 Partially supported by CIP/SW URI: Software Quality and Infrastructure Protection for Diffuse Computing (PI: Andre Seedorv, UPenn)

Learning network path information

- Network nodes (in the Internet or another network) need to know how to route data to a destination in the network
- In today's Internet, the Border Gateway Protocol (BGP) is used to share information about network paths
- BGP allows network instability
 - Lack of design guidelines to prevent bad interactions between separate local policies
- Can we give design principles for BGP-like protocols?
 - Prevent global instability
 - Can this be done using local conditions?
 - Identify other desirable protocol properties
 - Show trade-offs between these properties
 - Look at an important subclass of protocols

Design tradeoffs

- Are there local constraints that guarantee robustness?
 - Conjecture: No PVPs can exactly capture the set of robust configurations
 - Theorem: Systems in which extending paths increases their absolute rank are robust (smaller-ranked routes are preferred)
- The increasing condition may preclude other design goals:
 - Share information and check rank locally; lose autonomy and policy opaqueness
 - Let the protocol filter non-increasing routes; lose transparency
 - Have a mechanism collect all policies and check the increasing condition; generally intractable
- Main theorem about trade-offs:
 - A transparent, robust PVPs that supports autonomy of neighbor ranking and is at least as expressive as shortest paths must have a non-trivial global constraint

Centralized constraint check

- Given a routing instance, construct a directed graph whose directed cycles are potential dispute wheels; cycle check this graph
 - In network graph, replace each edge (u, v) with pair of directed edges (u, v) and (v, u)
 - Take these directed edges as the vertex set of a new directed graph; include the directed edge $((u, v), (w, x))$ iff $v=w$ and $P(C, C_j)$ holds, where v puts u in class C , and x in class C_j
 - Cycles in this graph are potential dispute wheels
 - If the network graph has maximum degree D and E edges, the running time is $O(D \times E)$

Protocol design goals

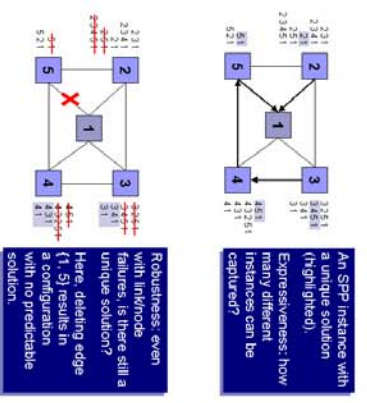
- Expressiveness: Can we write protocols and policies that capture many different routing configurations?
 - Use the Stable Paths Problem (SPP) of Griffin, Shepherd, and Willing to formalize expressiveness
 - How many SPP instances can we capture?
- Robustness: We want only routing configurations that are uniquely solvable, even with link and node failures
 - A routing protocol should be as expressive as possible without being able to express non-robust configurations
 - BGP is too expressive (can write non-robust configurations)
- Others
 - Transparency
 - Policy opaqueness
 - Autonomy
 - Global constraint

HBGP and class-based systems

- In Hierarchical BGP (HBGP), nodes classify neighbors as customers, providers, or peers based on business relationships
- Natural and simple restrictions then guarantee robustness (Griffin, et al)
 - E.g., don't share routes learned from one provider with another
 - Constraint: prohibit customer-provider cycles in the network
 - This is a global constraint naturally enforced by Internet economics
- Class-based systems generalize HBGP
 - Specify a set of classes to use in labeling neighbors
 - Require consistency between labels
 - Define relative preference rules
 - Require, e.g., that routes from neighbor in class C are always preferred to routes from a neighbor in class C_j
 - Define scoping rules
 - To neighbors in which classes may routes from a neighbor in class C , be exported?
 - Rules restrict what can be done without increasing level

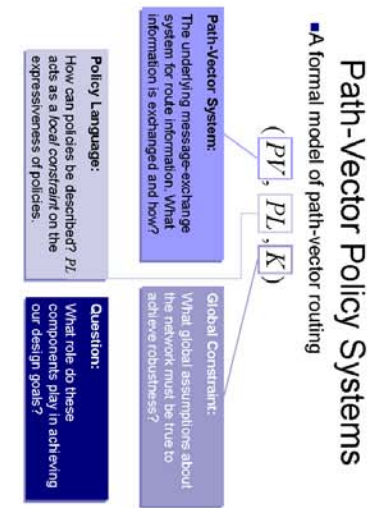
Distributed constraint check

- Token passing to add the network link between u and v
 - To check the directed link (u, v) , u sends fresh token to v
 - A node w receiving the token from x (labeled with C_j) forwards it to all neighbors z labeled with a class C , such that $P(C, C_j)$ holds
 - Merge token copies so that at most one such message per edge
 - If no such nodes, send a "clear" message back to x
 - If u receives a token from w that it would forward to v , the token has traversed the rim of a potential dispute wheel; u then sends a "clear" message back to w
 - Once w receives "clear" messages from every neighbor to which it forwarded the token from x , w sends a "clear" message to x
 - Use final "all clear" message to allow memory to be freed up
 - Either 0 or 3 messages per edge (original, "clear", and "all clear")
 - Should run this in the reverse direction, sending a token to u
 - What about parallel runs of the algorithm?



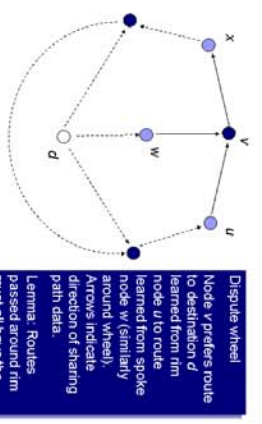
Class-based global constraint

- Dispute-wheel freeness is the broadest known sufficient condition for PVPs robustness
- Define a predicate on classes used to label neighbors
 - $P(C, C_j)$ is true iff a node might appear on a dispute wheel rim between neighbors i labeled with classes C_i and C_j
 - The node could be forwarding data (● in a rim segment) without increasing level
 - The node could prefer routes learned from neighbor on rim to neighbor on spoke (● where a spoke meets the rim)
 - Constraint: prohibit cycles in which every adjacent pair of edges satisfies this predicate
 - For HBGP, this reduces to preventing customer-provider cycles
- Any cycle that fails this constraint can be made into a dispute wheel (adding non-rim edges)
- Two different ways to check this global constraint



Path-Vector Policy Systems

- A formal model of path-vector routing



Conclusions

- Formalization of protocol design space
 - PVPs framework separates policy language from protocol
 - Identification of design goals
- Trade-offs between design parameters
 - Local constraints alone are not sufficient to obtain robustness and other design goals simultaneously
 - Global constraints are needed; what are good global constraints to use in practice?
- Class-based systems
 - Defined best-known constraint guaranteeing robustness
 - Checkable using centralized or distributed algorithm

