Efficient Algorithms For Optimizing Policy-Constrained Routing

Andrew R. Curtis
curtisa@cs.colostate.edu

Ross M. McConnell
rmm@cs.colostate.edu

Dan Massey
massey@cs.colostate.edu

***Abstract*— Routing policies play an essential role in how traffic is forwarded across the Internet. The network would not be commercially viable without these routing policies, but policies also introduce inefficiencies and fail to fully exploit the underlying network topology. Our work assumes routes are selected according to some policy such as a valley-free routing policy. However, we apply policy at an aggregate traffic level and don't require individual packets to follow paths that match the policy. Our approach never reduces, and usually increases, the connectivity and capacity of the network, and does not infringe on the underlying motivations that led to the routing policy. By adopting this approach, we also provide polynomial algorithms for otherwise NP-hard problems, including finding maximum policy-observing routing capacity between two sets of AS's, minimumizing cuts separating all policy-observing paths between two sets of AS's, and maximumizing sets of edge- or vertex-disjoint policy-observing paths.**

I. Introduction

In today's Internet, routes are selected based on a combination of network topology and routing policies. The network topology is a function of the physical connections and a number of studies have attempted to provide accurate models or estimations of the Internet topology. Given the network topology, one can answer basic graph theoretic problems such as what is the shortest path between a source and destination or how many disjoint paths exist between a given source and destination. But the topology is only part of the story. The set of possible paths between a source and destination depends on both the network topology and the routing polices at nodes along these paths. Some paths that exists in the topology may be disallowed due to routing polices. To identify the potential routes between a given source and destination, one must consider both the network topology and the routing policies used in the network.

As a simple example of routing policy, suppose Colorado State University has purchased Internet service from AT&T and Sprint. In the resulting network topology, there is a link connecting Colorado State to AT&T and a link connecting to Colorado State to Sprint. The path (AT&T, Colorado State, Sprint) exists in the topology and could theoretically be used to forward a packet from an AT&T customer to a Sprint customer. However, routing policy prevents an AT&T customer from using Colorado State's network to reach a Sprint customer. Even if the Colorado State network had sufficient resources to serve as an intermediate link between AT&T and Sprint, there is no economic incentive for Colorado State to provide this service. In fact, Colorado State purchased connectivity from AT&T and Sprint. AT&T cannot use its customer Colorado State as a transit network for other traffic. This simple example illustrates how policies can impact route selection. A comprehensive examination of Internet routing policies can be found in [Gao00]

Routing policies are an essential component of the Internet, but routing policies can also lead to both practical and theoretical limitations. In this paper, we develop new theoretical results about the impacts of policy on the vulnerability of a network to failure, the capacity and efficiency of a network, the efficiency of optimal routing algorithms. We extend the discouraging results of [EHPV05] to a broader class of optimization problems. Moreover, we show that the introduction of policy can lead to solutions that fail to be *pareto-optimal*, which is considered highly undesirable in economic theory, as it means that the costs for some actors can be reduced without increasing the cost for any of the actors, reflecting a needless expenditure of resources that is devoid of benefit for anyone. These results are not encouraging and would seem to suggest that routing polices have considerable practical and theoretical drawbacks.

However, our main result is to show that if a trivial change is made not to the policies themselves, but in minor details about the way they are executed, these negative impacts can be made to vanish. The key change is to enforce policies at an aggregate level rather than on a individual packet basis. Our work suggests that individual packets should be allowed to deviate from the routing policy. By allowing a packet to deviate from the routing policy, an individual node may be required to perform processing that would not have been necessary under the routing policy and a link may carry traffic that would not have been allowed under the routing policy. But the node or link is compensated for this cost by having some other packet deviate from the routing policy. Overall, no node or link is worse off than it would have been under the policy, but some nodes and links gain and the network as a whole is better off.

Our primary result is to demonstrate the power of such an approach and motivate work on routing protocols that follow what we call a currency-level approach to routing policies. Section II begin by formally introducing the concepts of routing policies and policy constrained network routing. Section III then shows the limitations of enforcing routing policies on a packet level basis. Section IV then introduces our currency-level approach to enforcing routing policies and

shows the how the limitations of packet-level approaches are reduced.

II. Policy Constrained Network Routing

We begin this section by formally defining what is meant by a routing policy and then formally define a network routing. Our work is motivated by routing policies that are used in today's Internet, but we also seek a model that is general enough to include a wide variety of routing policies. In particular, future Internet designs or other large-scale networks may devise policies unlike those in the current network and we would like our results to hold under a very broad class of potential routing policies. Toward this end, we first introduce the abstract notion of a routing policy and then show how this definition can be used to capture current Internet routing policy models and then formally define a network routing

A. Routing Policies

Let $G = (V, E)$ denote a network topology where V represents the nodes and E represents links between the nodes. For example, the nodes in V may represent routers or may represent larger entities such as an Internet Autonomous System. It does not matter how the topology was derived and we do not assume that any node or any central authority knows the full topology. The routing policy at node $x \in V$ is defined as follows:

Definition 1: A **routing policy** at node x is a set of pairs $\{(E_1, E'_1), (E_2, E'_2), \ldots, (E_k, E'_k)\}$, where $\{E_1, E_2, ..., E_k\}$ is a partition of incoming edges into x, and where a packet coming into x on an edge E_i can be forwarded through any edge in E'_i. The sets $\{E'_1, E'_2, ..., E'_k\}$ are distinct but need not be disjoint, and some E'_i can be empty, meaning that x cannot forward packets coming in on an edge in E_i.

This general model is intended to cover a wide variety of potential polices and explictly allows different policies at different nodes. For example, node x may represents an AS and our model allows each AS to independently selects its own routing policy. To see how this model applies in the Internet, we show how the model can be used to create a *valley-free routing policy*.

In the Internet, routing policies often reflect customer-provider relationships between Autonomous Systems. That is, given two ASes which are connected by a link, one AS is a customer of the other AS which must be the provider. In the valley-free model, a packet can never go from a provider to a customer and then back to a provider again. Such a "dip" in the path is called a **valley**. Erlebach *et al.* [EHPV05] formalize this customer-provider relationship by modeling the AS graph as a directed graph where edges "point upwards" from customers to providers, that is, if (x, y) is an edge in the graph G, then x is a customer of y, and we do not allow the anti-parrallel edge (y, x) to also be an edge in G. A path in this model is said to be valley free if it leads up from customers to providers and then decreases from providers to customers.

To capture this in our model, each vertex has two policies: one for packets coming from customers and siblings, and one for packets coming from providers and peers. More formally, the two policies at each node are (E_1, E'_1), and (E_2, E'_2), where E_1 is the set of edges coming from providers and E'_1 is the set of edges that can be used to forward packets that arrived from a provider. Similarly, E_2 is the set of edges coming from customers, and E'_2 is the set of edges that can be used to forward packets that arrived from customers. Enforcing these constraints results in a **valley-free model**, where we can view the restrictions placed on the packets flowing to each outgoing edge of a node as policy.

B. Network Routings

In this section, we formally define the concept of policy constrained network routing and routing policies.

Definition 2: Let us initially define a the parameters to a **policy-constrained routing problem**, as follows. The inputs are the following:

1) A *network*, which is an undirected graph representing the communication links;
2) A *destination vertex* t, a set $s_1, s_2, ..., s_p$ of *source vertices* generating packets destined for t, and integer rates $i_1, i_2, ..., i_p$ reflecting the rates at which these packets are generated;
3) A (possibly unbounded) *capacity* on each edge, indicating the maximum rate at which packets can be routed through the edge or the vertex (useful for solving edge-disjoint path problems);
4) A *cost per packet* on each vertex, reflecting the cost of routing a packet through that vertex;
5) A set of policies at each vertex v, subject to the condition that no two policies at v share an incoming edge.

And finally, the following gives the definition of the problem we would like to solve on policy-constrained routings.

Definition 3: The following are defined in terms of the inputs to the policy-constrained routing problem.

- A **packet routing** is the set of paths, each from some source s_i, to t, where the path is labeled with the rate at which packets follow that path. There can be more than one path from a single path s_i to t. The sum of rate labels of the paths is the rate of flow from $\{s_1, s_2, ..., s_p\}$ to t (no packets are unaccounted for).
- A routing path is **feasible** if it violates no edge capacity and no packet's path violates a policy at any node that it traverses.

III. Limitations Packet-Level Routing Policies

In the above model, the path taken by a packet to does not violate the routing policy at any node. We will refer to this conventional enforcement of policies as **packet-level enforcement**. In this section we show that requiring every packet to obey the policy results in a number of negative consequences. Section III-A shows that current routings in the Internet are not optimal when viewed under economic theory and Section III-B then explores the computational properties of optimization problems. The packet level constraints result in both practical and theoretical limitations.

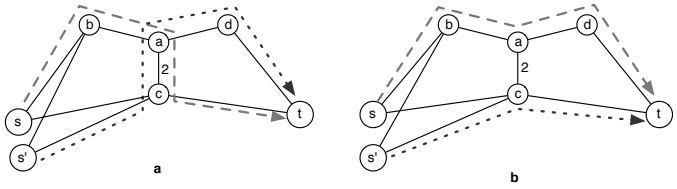

Fig. 1. In both figures, lower-level nodes are customers of higher-level nodes, ie. s is a customer of b and c. And, all edges have capacity 1 except for ac which has capacity 2. The first figure (a) depicts a valley-free flow from sources s and s' to a destination node t. (An edge that goes from a lower to a higher node in the figure represents an edge from a customer to a provider.) The second figure (b) illustrates an improvement obtainable if a and c are allowed to trade identities of packets that they handle through swapping of packet "envelopes."

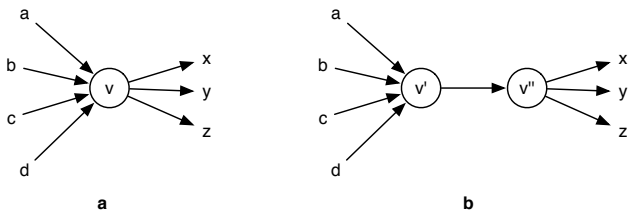

Fig. 2. The construction of Theorem 2. Figure (a) shows the original node v, and (b) shows v split into two nodes so that we can constrain the amount of flow through it. The edge (v', v'') would be labeled with the amount of flow through v.

A. Policy Constrained Routings are Not Pareto-Optimal

Routing policies are typically motivated by economic concerns, but the results often fail to be **pareto-optimal**. A set of transactions among a set of actors that satisfies a set of economic constraints is pareto-optimal if every other feasible transaction that is preferred by any actor is strictly less preferable to at least one other actor. In other words, the transactions are pareto optimal if no actor can be made better off without making some other actor worse off. Transactions that are not pareto-optimal are considered highly undesirable in economic theory: they can be modified to increase the total benefit of the transaction to the actors at no additional cost to any of them, and therefore waste resources.

Theorem 1: It can be the case that no feasible packet routing is pareto-optimal.

Proof: An example for the case when the policy is the valley-free constraint at all nodes is depicted in Figure 1. Suppose one packet per time unit must be routed from s and s' to t and all edges have a capacity of one, except for ac, which has a capacity of two. All solutions that enforce the policy at the individual packet level must route one unit of flow through each of b and d and two units through each of a and c. The solution depicted in the (b), which routes one unit of flow through each of these four nodes without increasing the flow through any node, is pareto-optimal however. ■

Our next result shows that pareto-optimal routing can be found in polynomial time. Despite this, the packet constraints prevent such a routing from being valid. In Section IV, we show how to gain pareto-optimal routings by relaxing the constraint that policy be enforced on each individual packet.

Theorem 2: Given any set of policies and any feasible routing, a pareto-optimization that minimizes the sum of flow costs at the vertices can be found in polynomial time.

Proof: The proof is by reduction to the min-cost flow problem [Chv83], where each node v is replaced by two nodes, v', which takes becomes the head of the edges into v, v'', which becomes the tail of all edges out of v, and an edge (v', v''). The capacity of edge (v', v'') is the flow through v given by the routing, and the capacity of each edge is the flow through the edge given by the routing. This construction is shown in Figure 2. The cost of edge (v', v'') is set to the per-unit cost of flow through v, and the capacities and costs of edges of the form (v'', w') (edges in the original graph) are set to infinity and 0, respectively. The capacity constraint on edge (v', v'') ensures that no more flow is routed through a node v than in the original solution. The fact that the sum of costs of flows through the nodes is minimized means that no flow can be reduced through any node without violating a capacity constraint, that is, the constraint that more flow be routed through some node than in the original routing. ■

The limitations of applying policy on a packet level basis creates suboptimal scenarios such as the ones described above. In the next section, we consider how the packet level policies impact theoretical results and optimization problems.

B. Optimization Problems and Policy

Two paths in a graph are edge-disjoint if they have no edges in common, and vertex-disjoint if they have no vertices, except their endpoints, in common. Vertex-disjoint paths are edge-disjoint, but the converse is not necessarily true. The rate at which packets from a source node to a destination node flow on different links is known as a **flow**. The **capacity** of a link or node is the maximum rate at which it can route packets. If s and t are nodes of a network, an **s-t cut** is a partition of the nodes into a set S containing s and a set T containing t. The **capacity of the cut** is the sum of capacities of the edges from S to T.

A common measure of the robustness of communication from s to t in a networks is a **minimum-cardinality edge cut**, that is, a minimum-sized set of edges that must fail (or be sabotaged) in order to disconnect all paths from s to t. What is of interest in studying routing and robustness and routing topology of the Internet, however, is the *number* of edges that must be removed in order to cut all policy-satisfying paths from s to t, since the policies at the nodes dictate that this will render t inaccessible from s.

A related problem, which occurs in many applications is one where a high premium is placed on the timely arrival at t of a message from s. The strategy of s is to decide in advance on a set of edge disjoint paths to use in sending messages from s to t. The paths are chosen to be edge-disjoint so that no one edge failure can disrupt any two paths; s seeks a maximum set of such paths. In the absence of policy constraints, a maximum

set of edge-disjoint paths from s to t can be found efficiently using network flow algorithms.

Clearly, the number of paths in a maximum set $\mathcal{P}$ of edge-disjoint paths from s to t is a lower bound on the size of a minimum cut, since a minimum requirement of any s-t cut, which must cut all paths from s to t, is that it must cut every path in $\mathcal{P}$. Since the paths in $\mathcal{P}$ are edge-disjoint, an s-t cut must contain at least one edge from each member of $\mathcal{P}$.

A famous theorem in graph theory, known as **Menger's theorem** [Wes01] is that the number of edge in a min s-t cut is *equal* to the maximum number of edge-disjoint paths from s to t. According to this theorem, if there are no policy constraints, s may commit to a set of stable edge-disjoint paths in advance, and an intruder who knows these paths is not helped by this knowledge in deciding on a set of links to undermine in order to halt communication from s to t; whether or not s is committed to a particular maximum set of edge-disjoint paths, the intruder must undermine a set of edges that cuts *all* s-t paths in order to cut this set of paths.

Erlebach *et al.* investigate several of the these classic problems in the valley-free path model, and find they are no longer valid in the valley-free model. Their results show that valley-free policy has a striking effect on the robustness and connectivity of the internet and are summarized as the following [EHPV05]:

1) Menger's theorems no longer apply when policy constraints are introduced. In particular, under the valley-free model, the maximum number of edge or (vertex) disjoint paths from s to t can be as small as half the minimum number of edges (vertices) whose removal disconnects all feasible s-t paths.
2) Finding the minimum number of edges (vertices) whose removal disconnects all feasible paths from s to t is NP-hard, as is finding the maximum number of edge-(vertex-) disjoint paths from s to t.
3) There exists an efficient approximation algorithm that finds a set of edge disjoint feasible paths whose number is at least half the maximum number, and an approximation algorithm that finds a set of edges whose removal disconnects all feasible paths that is at most twice as large as the minimum number required.
4) Unless P = NP, no better worst-case approximation can be found in polynomial time.

While Erlebach *et al.* showed that the robustness and connectivity of the internet is negatively impacted by policy, they did not show that policy impairs the network's ability to find flows that maximize the utilization of the links between s and t. We will now extend their results to network flows.

Let a **maximum feasible policy s-t flow** be a maximum flow from s to t that respects the network's packet constraints, ie. in a valley-free flow all paths in the flow can never contain a valley. In a network setting, the maximum feasible policy s-t flow is an upper bound on the amount of traffic that can flow from s to t if policy is respected. In classical graph theory, it's well known that the minimum edge cut is equal to the maximum flow. This is the celebrated min-cut, max-flow theorem [Chv83]. We show that the min-cut, max-flow theorem no longer holds when policy is enforced at the packet level in Theorem 4. Before showing this, however, we show that the addition of policy makes it intractable to compute the maximum feasible policy flow between two nodes, assuming P $\neq$ NP.

Theorem 3: It is NP-hard to find a maximum feasible policy s-t flow under the packet model, and this is true in the special case of valley-free flows.

Proof: Erelebach, *et. al.* show that it is NP-hard to find a maximum set of edge-disjoint valley-free paths from a vertex s to a vertex t in a graph where the customer-provider and peer-to-peer relationships are marked on the edges. Therefore, the question of whether there exists a set of k edge-disjoint valley-free valley free paths on such a graph is NP-complete. However, this question can be solved in polynomial time if there exists a polynomial-time algorithm for the path-constrained routing problem, as follows. Give each node two policies reflecting the valley-free constraints (one for customers and siblings and one for providers and peers), assign a capacity of one to each edge, let s be the only source vertex and t be the destination vertex, and assign s an output rate of k. This policy-constrained routing problem is feasible if and only if the given problem has k edge-disjoint valley-free paths. ■

The following shows that policy can lower the overall throughput of a network.

Theorem 4: Given a network with arbitrary edge capacities and policies at the nodes, the maximum feasible policy s-t flow under the packet constraints can be strictly smaller than the minimum-capacity policy cut, and this is true in the special case of valley-free flows.

Proof: (Sketch) Erelebach, *et. al.* show that the number of paths in a maximum set of edge-disjoint paths can be as small as half the number of edges in a minimum policy edge cut. The reduction similar to that of Theorem 3: for a network where the minimum cardinality edge cut is $2k$ and the maximum number of disjoint paths is k. Establish a corresponding flow problem where the edges all have capacity equal to one. A maximum set consisting of k edge-disjoint paths constitutes a maximum packet-constrained flow of size k, and a minimum policy cut constitutes a policy cut of minimum total capacity, which is $2k$. ■

While our theorem is a theoretical result, we claim that it is applicable to the real-world Internet. For instance in [GW02], the degree to which the lengths of routing paths are increased due to policies, called *path inflation*, is studied and evidence is provided that it is significant.

Thus far, we have pointed the finger at policy for all of our negative results; however, the negative computational and economic effects we have shown do not arise as a result of policy. Instead, *these effects arise as a result of enforcing policy at the packet level*.

We next propose a new method of policy enforcement that allows ASes to keep the ability to specify arbitrary policy, but

does not have any of the negative effects we have shown here that arise when policy is enforced at the packet-level.

IV. Currency-Enforcement of Policy

By *packet constraints*, we mean the constraints that an individual packet must obey on a valid path. This definition, which matches the way policy is currently enforced on the Internet, implies that each packet has an individual identity. To alleviate the negative effects of packet-level enforcement, we remove this notion of packet identity. That is, by the **currency constraints**, let us denote the constraints that must be obeyed when one is allowed to find a solution that obeys the packet constraints, and then reduce the flow through some nodes without increasing the flow through any node by ignoring the identities of individual packets.

As an example of the currency constraints, consider the two flows shown in Figure 1. Assume that both flows are equal in size. In Figure 1(a), the nodes a and c route packets between each other because their policies dictate how each packet from the previous hop must be routed, ie. in the case of c, its policy dictates that any packet arriving from s' must be routed to a if ct is saturated. Let's consider what happens under the currency constraints. Nodes a and c notice[1] that they can "cancel" out flow between them by violating the packet constraints. Both nodes reduce the flow through them by enforcing policy at a currency-level rather than at the packet-level. In Figure 1(b), no node has an increased amount of flow going through it, but nodes a and c are better off since they each have less flow going through them.

We can now achieve routings that are pareto-optimal with respect to the costs are each node. Once again, consider the example given in Figure 1. It's clear that the flows in Figure 1(a) create costs that are not pareto-optimal; however, under the packet constraints, there is no other way the packets can be routed (recall that edge ct has capacity 1). As just discussed, with the currency constraints the flow shown in Figure 1(b) is now valid. Thus, the currency constraints allow the routing to become pareto-optimal. Furthermore, this pareto-optimization was done with only an agreement between a and c. We claim that, in general, pareto-optimizations under the currency constraints can be found with only pairwise agreements between nodes.

In order to prove routing results about the currency constraint model, we need to construct a graph that captures all policy information of the network we are given, the next section gives this construction.

A. Policy Constraint Graph

For a policy-constrained routing problem, we say its **policy constraint graph** is a directed graph where each directed path in the policy constraint graph is a valid path in the policy-constrained routing. We'll now give constructions of the policy constraint graph in the cases where the edges have unbounded capacities and then for the slightly trickier case when the edges have bounded capacity.

[1]For now, we assume that the nodes have a mechanism to detect this optimization. We discuss this further in Section VI

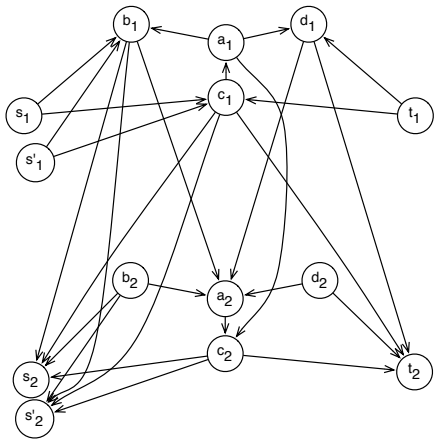

Fig. 3. The unbounded policy constraint graph of the network shown in Figure 2.

For now, let's assume that all edges in the given network have unbounded capacity. Let the policies at x be $\{(E_1, E'_1), \ldots, (E_{k_x}, E'_{k_x}\}$. Then, in the **unbounded policy-constraint graph**, x is split into k_x vertices, say $x_1, x_2, \ldots, x_{k_x}$. Vertex x_i then has incoming edges E_i and outgoing edges E'_i.

As a simple example, Figure 3 shows the unbounded policy constraint graph of the example network in Figure 1 where the network follows the valley-free policy model. In Figure 3, it's clear that once a path has a provider to customer edge, it can never go back "up" to a provider.

In order to construct the more general **bounded policy constraint graph**, let the capacity of an edge xy be $c(xy)$. As before, let the policies at x be $\{(E_1, E'_1), \ldots, (E_{k_x}, E'_{k_x}\}$, and let $x_1, x_2, \ldots, x_{k_x}$ be the vertices representing each policy of x. For each edge $x_i y_j$ in the unbounded policy constraint graph, where $1 \leq i \leq k_x$ and j is fixed, we need to insert a "dummy" node that allows us to bound to overall flow from all x_i's to y_j. So, we redirect the edges from each x_i to y_j so that they all point to a new node d_{xy}, and we give them an unbounded capacity. Then, in order to retain the capacity of xy in the original network, we add the edge $d_{xy}y_j$ with capacity $c(d_{xy}y_j) = c(xy)$. This edges limits the amount of flow that can travel from any x_i to y_j but does not specify how the flows should be distributed amongst the various outgoing policies of x (hence it does away with the notion of an individual packet identity which is precisely what we defined the currency constraints as doing).

An example of a bounded constrain graph is shown in Figure 4. There, each vertex has three policies, so the constraint graph has three versions of each vertex. Note how xzy is a path in the original network on in (a), and it is still possible to get from x to y in the constraint graph shown in (b), but the path reflects the policies of each node.

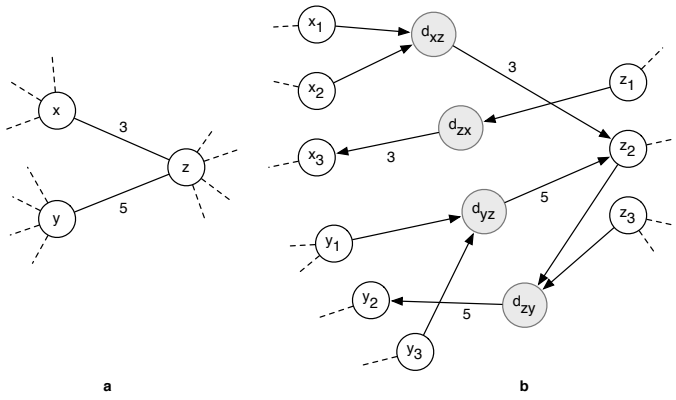

Fig. 4. Incorporating edge capacity constraints into the unbounded policy constraint graph. In figure (a) are two edges xz and yz of a network with capacities 3 and 5 respectively. Each vertex has three policies: xz is an output edge for x's first two policies; zx is an output edge for z's first policy; yz is an output edge for y's first and third policy; zy is an output edge for z's second and third policy. Figure (b) shows how these constraints are modeled in the bounded constraint graph: Each dummy node d_{uv} has one outgoing directed edge that enforces the capacity constraint of the edge uv in the network. The directed edges from copies of a node u to a dummy node d_{uv} enforce the policy constraints; there is no such edge from copy u_i if uv is not an outgoing edge for u's i^{th} policy.

B. Currency-Constrained Routing Results

The following theorems show that the discouraging computational results of Section III-B about policy disappear when policy is enforced at the currency-level rather than the packet level.

Theorem 5: Under the currency constraints, it takes polynomial time to find an s-t flow that is as large as the minimum s-t policy cut, and the algorithm can be implemented to run in polynomial time on a distributed model, using only pairwise agreements between adjacent nodes, where each of the two parties benefits from the agreement.

Proof: Let a *type 1* edge in the bounded policy constraint construction be one that goes from a dummy vertex node to a copy of a network node, ie. Figure 4 has four type 1 edges $d_{xz}z_2, d_{zx}x_3, d_{yz}z_2$, and $d_{zy}y_2$. And let a *type 2* edge be one that goes from a copy of a network node to a dummy node. The type 2 edges have unbounded capacities, so a minimum-capacity cut consists exclusively of type 1 edges.

If a set of edges go from a set of copies $x_1, x_2, ..., x_k$ of a node x to dummy vertex, let us denote the dummy vertex by x'. Let the *image* of a type 1 edge (x', y_j) in the network be the link xy that gave rise to it in the construction. Note that (x', y_j) and (y', x_h) both map to the same image for some $1 \leq h, \leq k$, where k is the maximum number of policies. Because j is the unique policy at y for packets coming into y on xy and h is the unique policy at x for packets coming into x on xy, *these are the only two edges that are inverse images of* xy.

Let s_i denote the copy of s that corresponds to the policy for packets originating at s in the network. A path from s to t is a policy-respecting path if and only if it is the image of type 1 edges on a path from s_i to a copy t_j in the constraint graph. By a min-capacity s-t path in the constraint graph we denote a minimum edge cut that severs all such paths in the constraint graph. Suppose that this has a total capacity of c. The image of this cut in the constraint graph has capacity at least $c/2$, since each edge of the image is the image of at most two edges of the cut. This cuts the images of all paths from s_i to t in the constraint graph, so it cuts all policy paths in the network.

By the max-flow min-cut theorem, there is a flow from s_i to copies of t of magnitude c. The image of this flow is a flow in the network where a link xy of capacity c_i can have a flow of at most $2c_i$ on it: at most c_i units from x to y and at most c_i units from y to x. The currency model allows the minimum of these two flows to be canceled by an equal portion of the flow in the other direction, yielding a flow of at most c on the edge. Since this resolution can be performed at all edges of the network where the flow exceeds the capacity, the final flow satisfies the capacity.

Each resolution of a capacity violation is an optimization permitted under the currency model, but not under the packet model. Moreover, each resolution requires an agreement between two adjacent nodes x and y, and the agreement reduces the routing costs for both x and y. ■

Theorem 6: It takes polynomial time to find a set of edge disjoint paths under the currency constraints that is as large as the number of edges in a minimum policy edge cut, and the algorithm can be implemented on a distributed model using only pairwise agreements between adjacent nodes, where each of the two parties benefits from the agreement.

Proof: (Sketch) The proof is by reduction to Theorem 5, by assigning unit edge capacities to the links of the network. ■

V. New Algorithms for Packet-Constrained Optimization Problems

In this section, we once again consider optimization problems under the packet constraints. Now armed with our powerful policy constraint graph construction, we are able to find an efficient algorithm to find shortest policy-observing paths in Theorem 7. And finally in Lemma 1, we use our construction to find a k-approximation algorithm for the edge-disjoint policy paths, where k is the number of policies at each node.

Theorem 7: Given a network with n nodes and m links and at most k arbitrary policies at each node of the form given in Definition 2, point 5 at each node, it takes $O(k(n+m))$ time to find the shortest policy-observing paths from every node to a given destination node.

Proof: The construction of the unbounded constraint graph creates at most k copies of each vertex, so the number of vertices is at most kn. The key to bounding the number of edges is the fact that the incoming edges for different policies at a node v are disjoint. Thus, for any edge uv, there is at most one copy v_j of v that receives incoming directed edges from copies of u. An edge uv of the network maps to at most

k edges of the form (u_i, v_j) and at most k edges of the form (v_h, u_l), for a total of at most $2k$ edges.

Let s_i be the copy of s in the graph that corresponds to the policy for packets originating at s. Let the image of an edge (x_i, y_j) in the network be link xy. A path in the network is a policy path if and only if it is the image of the sequence of edges in a path from s_i to a copy of t. Run a breadth-first search from s_i. For each node x, find the copy x_i of x that has a minimum distance label, and the parent y_j of this node in the breadth-first search tree; let the *parent* of x in the network be node the node y that is the image of y_j. The parent relation is a tree rooted at s such that the unique tree path from s to any node x is a shortest policy path. ■

A somewhat tighter bound is $O(k_1 n + k_2 m)$, where k_1 is the sum, over all vertices of the number of policies at the vertex, and k_2 is the sum, over all edges, of the number of policies at the endpoints of the edge.

Lemma 1: There is a k-approximation algorithm for finding edge-disjoint policy paths under the packet constraints, given a network with at most k policies at each node.

Proof: (Sketch) Let the image in the network of a vertex x_i in the unbounded policy graph be the node x that gave rise to it in the construction. By an argument analogous to the one in the proof of Theorem 7, a set of vertices is an s-t policy cut in the network if it cuts all paths from the copy s_i of s that handles packets originating at s from all copies of t. The image such a set in the network is at least $1/k$ times as large, since each node of the network is the image of at most k vertices of the cut. ■

The result of Erlebach *et. al.* is therefore a special case of our Lemma, reflecting the fact that valley-free consists of two policies at each node, one for providers and peers, and one for customers and siblings. Erlebach *et. al.* also show that, unless P = NP, no better approximation bound is possible, but we do not yet know whether this result generalizes to Lemma 1.

VI. Discussion

The results presented here show we can gain a great deal by enforcing policies on a currency-level rather than on a packet level. This suggests a new direction in developing routing algorithms. A traditional routing algorithm such as BGP computes networks routes according some routing policy. Once the routes have been computed, a second phase monitors traffic and looks for optimizations where packets may be forwarded on paths that do not comply with the policy. These optimizations can be performed on a localized basis and need not require cooperation of all nodes in the network. Nodes which fail to cooperate continue to use the paths computed by the traditional protocol such as BGP. Nodes which choose to cooperate can gain by agreeing to diverting packets onto non-policy compliment routes in return for similar actions by other nodes in the network.

One possible protocol for taking advantage of the currency relaxation involves advertising of paths. Senders send groups of k packets accompanied by a small **envelope** that indicates the paths that the packets are expected to follow. One may set k to be large to make the cost of forwarding envelopes trivial compared to the cost of forwarding packets. The actors agree that packets cannot proceed without an accompanying envelope, but that it is acceptable to swap the contents of two envelopes that are bound for the same destination. One consequence is that, after optimization, some envelopes may travel empty for some portions of their path.

Work remains to explore practical implementations of these protocols, as well as to find distributed algorithms that make optimal use of them. One immediate concern is packet looping. A traditional routing protocol such as BGP computes the policy compliant routes and prevents the formation of long lasting routing loops. But our approach allows a packet to deviate from the policy compliant route and thus loop freedom is no longer guaranteed. We first argue that any loops will not cause problems at the aggregate traffic level. Since the improvement is an optimization of a flow that is indistinguishable from one that observes the policy at the level of the packets, there is no danger that cycling of packets will increase the traffic in the network.

But this guarantee is of little comfort to a packet that may become permanently stuck in loop. One approach under the currency model is to take advantage of nondeterminism in deciding which of two indistinguishable packets at a node to route to two distinct next hops. This makes the probability of a packet cycling k times vanish exponentially as k increases. A second more deterministic approach is keep a count of how many times a packet is diverted. The sender sets an initial value in a $DivertsLeft$ field and this field specifies how many times the packet may be diverted from onto a non-policy compliant route. $DivertsLeft$ is decremented each time the packet is forwarded out an interface that does not match the interface of the policy compliant route. When $DivertsLeft$ reaches zero, the packet can no longer be diverted from the policy compliant route. Since the policy compliant route is assumed to be loop free, the packet is now forward along this loop free path.

VII. Conclusions and Future Work

This paper considered the impact of routing policy and showed how enforcing routing policies on a packet level results in several practical and theoretical limitations. The problem however is not with the routing policies, but rather with the policies are enforced. The problems occur when policies are enforced on each packet. By instead taking a currency level approach to enforcement, one can still meet the broad goals of the routing and eliminate the problems. These results suggest that future work on routing protocols should consider adopting a currency approach where individual packets can violate the routing policy as long as the overall system still meets the policy.

References

[Chv83] V. Chvatal. *Linear Programming*. W. H. Freeman, New York, 1983.

[EHPV05] T. Erlebach, A. Hall, A. Panconesi, and D. Vukadinovic. Cuts and disjoint paths in the valley-free path model of internet bgp routing. *Lecture Notes in Computer Science*, 3405:49–62, 2005.

[Gao00] L. Gao. On inferring autonomous system relationships in the internet, 2000.

[GW02] L. Gao and F. Wang. The extent of as path inflation by routing policies. *IEEE Global Internet Symposium*, 2002.

[Tar83] R. E. Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Math., Philadelphia, 1983.

[Wes01] D. B. West. *Introduction to Graph Theory*. Prentice Hall, Englewood Cliffs, New Jersey, 2001.