Reduced State Routing in the Internet *

Ramakrishna Gummadi †
Ramesh Govindan‡
Nupur Kothari§
Brad Karp ¶
Young-Jin Kim‖
Scott Shenker**

ABSTRACT

In today's Internet core, routers store forwarding state proportional to the number of edge networks. As the Internet grows and core line rates increase, routers require memories that are increasingly fast and large—and are correspondingly increasingly expensive and difficult to engineer. In this paper, we present Reduced-State Routing (RSR), in which core routers require state only concerning the network topology within a two-hop radius, and thus of a size independent of the total number of Internet edge networks. RSR achieves this feat by routing *geographically* using two sets of node addresses: *virtual coordinates,* that are assigned to reflect the link costs within an autonomous system; and *geographic coordinates,* that correspond to nodes' physical locations. RSR routes greedily on virtual coordinates, and falls back to *face routing* on geographic coordinates when greedy progress is impossible on virtual coordinates. Unlike previous geographic routing schemes, RSR works on Internet-like graphs (rather than only on wireless-like graphs), and supports policy routing. By simulating RSR on real tier-1 ISP topologies, we demonstrate that RSR achieves low path stretch, comparable to that caused by policy routing in today's Internet.

1. INTRODUCTION

With increasing aggregate router bandwidths in ISP cores, forwarding packets at line speed continues to be a major challenge for vendors [6]. One of the bottlenecks is IP lookup; the need to perform longest-match prefix lookups over large (and continually growing) routing tables has pushed researchers to consider CAM-based hardware solutions and innovative table compression schemes [5, 6, 17] that minimize memory bandwidth requirements and size. In this paper, we describe *Reduced-State Routing* (RSR), a routing architecture that greatly reduces the state requirements for packet forwarding at core Internet routers, and thus reduces the cost of core router hardware, both for high-speed forwarding engines, and for high-speed memory used to hold forwarding tables.

RSR builds on the geographic routing schemes discussed in the wireless routing literature [1, 10, 12]. In geographic routing, nodes are addressed by geographic location and nodes greedily forward packets to those neighbors closer to the destination than themselves. When no such neighbor can be found, geographic routing relies on *face routing* on a planar subgraph of the full network graph to escape this local maximum; see [1, 10, 12] for details. At each hop, nodes base all forwarding decisions solely on the positions of their neighbors and the position of the packet's destination. Thus, nodes' forwarding tables are of size equal to their fanout, and independent of total network size.

This scaling property has much appeal for Internet routing[1], but there are two drawbacks that prevent geographic routing from being directly applied in the Internet.

The first issue is that of topology. Existing geographic routing schemes rely on face-routing techniques to reach all destinations successfully. Face routing assumes radio-like connectivity, where nodes can directly communicate if and only if they are within a certain distance of each other. Internet connectivity does not have this property: connectivity is only loosely correlated with, not determined by, proximity. In recent work [11], we presented a distributed probing scheme called CLDP that guarantees correct face routing on *arbitrary* graphs. RSR employs CLDP to apply geographic routing to Internet-like graphs.

However, Internet routing requires more than mere reachability; policy constraints and traffic engineering (TE) considerations require choosing routes by metrics other than ge-

†University of Southern California. gummadi@usc.edu
‡University of Southern California. ramesh@usc.edu
§University of Southern California. nkothari@usc.edu
¶CMU and Intel Research. bkarp@cs.cmu.edu
‖University of Southern California. youngjki@usc.edu
**ICSI and UC Berkeley. shenker@icsi.berkeley.edu

*This material is based in part upon work supported by the National Science Foundation under Grant No. 0330178. Any opinions, findings and conclusions or recomendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

[1]Finn [7] and Deering [3] proposed geographic routing and addressing for the Internet, respectively. These approaches have much charm, but the former floods packets when a sequence of hops monotonically closer to the destination does not exist, and the latter requires that providers interconnect at all major metropolitan areas. Here we accept the current Internet addressing scheme and provider interconnection configuration as given, eschew flooding, and seek to achieve reduced routing state within those constraints.

ographic distance or hop count. Thus, the second problem with geographic routing schemes is that they slavishly follow the paths determined by node coordinates, and thus cannot easily accommodate intra-ISP TE and inter-ISP policy routing.

This paper focuses on this second issue, proposing in RSR a design that allows the forwarding state size in core ISP routers to be a function of node fanout, not network size, but that still provides the same administrative autonomy, policy hooks, and TE controls found in the existing Internet. RSR requires longer packet headers than IPv4's, but comparable in length to those used in IPv6. In terms of route length, on the real ISP topologies that we have tested, a preliminary version of RSR, without extensive optimization, achieves path stretches less than 2. To put this number in context, evidence suggests that policy routing in today's Internet inflates paths by that much or more [18].

We continue in Section 2 by describing RSR in detail. We then offer a performance evaluation of RSR in Section 3, and conclude in Section 4 with a discussion of future work directions.

2. REDUCED-STATE ROUTING

In this section, we discuss the design requirements for RSR, then outline the approach we adopt to meet those requirements. We then describe RSR's components in detail, and conclude by illustrating how they fit together with an end-to-end packet forwarding example.

2.0.0.1 *Design Requirements.*

Our central aim in RSR is to allow an ISP's core routers to require no forwarding table state beyond their immediate neighbors in that same same ISP's core. Traditional geographic routing allows each node to route using a forwarding table consisting only of one entry for each of its single-hop neighbors [10]. We add three other requirements for RSR crucial for use in the Internet that are unsatisfied by prior geographic routing schemes.

First, *RSR must choose intra-domain routes that reflect the ISP's traffic engineering decisions* **(R1)**. These routes often do not minimize Euclidean distance between source and destination, nor hop count. In this paper, we consider one way in which ISPs today engineer their backbones—by tuning link weights in their IGP (*e.g.*, OSPF or IS-IS).

Second, *RSR must find inter-domain paths that reflect ISPs' policy decisions* **(R2)**. Again, these AS paths, found today by BGP, typically do not minimize Euclidean distance, nor hop count.

While achieving goals R1 and R2, *RSR must allow ISPs to use the same "knobs" they do today to accomplish intra-domain TE and inter-domain policy routing* **(R3)**. To that end, we allow ISPs to continue to configure BGP policies and IGP link weights exactly as they do today. RSR fully preserves the control and policy planes of entities underpinning the current Internet's naming and routing system: Autonomous Systems, the two-level routing hierarchy consisting of intra-domain and and inter-domain routing protocols, and the Domain Name System. However, objects used in the data plane, such as node addresses, data used for computing routes within and across ISPs, and entries in the forwarding tables of routers' line cards are slightly different. [2]

2.0.0.2 *RSR Overview.*

We now give a schematic overview of RSR. Let us begin by considering intra-domain routing, inside a single ISP backbone's AS. A packet arrives at an ingress router at the edge of the AS. Let us assume for the moment that the egress router in that AS is known.

The ISP runs the same link-state IGP it does today, with edge weights that cause TE to occur. As a result, each router holds the full link-state database, which is the full intra-domain network graph within the AS. RSR does *not* generate a forwarding table from this information. Rather, it *embeds the network graph into an n-dimensional coordinate space, yielding n-dimensional virtual coordinates (VC) for each router.* This embedding has the property that the Euclidean distance between two nodes' VCs is proportional to the cost of the path between them in the network graph (*i.e.,* the sum of its link costs).

Within an AS, RSR simply routes greedily on these VCs. That is, each packet contains a VC destination address, each router knows its own VC address, and each router knows the VC addresses of its neighbor routers. A router computes its own Euclidean VC distance from the destination, and those of its neighbors, and forwards the packet to the neighbor closest to the destination, provided the neighbor is closer than the forwarding router.

Note that greedy routing on VCs may fail; it's possible that a router's neighbors may have greater or equal Euclidean VC distance to the destination than itself. In this case, RSR recovers by routing using a *different* set of addresses. Each packet also carries the *geographic* coordinates (GC) for the destination, its two-dimensional real-world position. And each router knows its own GC, and those of its single-hop neighbors. When greedy forwarding fails using VCs, RSR instead forwards the packet using GCs, by face routing atop CLDP, which *always* succeeds in any network graph [11].

The intuition behind this overall intra-domain routing approach is that greedy forwarding on VCs respects TE constraints, but may encounter regions of a topology where greedy forwarding is impossible. Face routing on GCs does not respect TE, but always succeeds on any topology. Thus, RSR tries VC routing first, and falls back to GC routing only when needed to ensure reachability.[3] The aim is that the resulting combination ensures reachability, with some (hopefully small) divergence from TE constraints.

[2]While incremental deployment of RSR bears consideration, we do not do so herein.

[3]We use GC's and not VC's for face routing because the face routing we currently use is guaranteed to find a loop-free path only in two dimension graphs.

We now consider inter-domain routing. An RSR sender marks all packets with a *policy tag* (PT), which identifies the destination host's AS. All ISP *edge* routers run BGP as they do today, but exchange routes with PT's (AS numbers) as destinations, rather than with IP prefixes. An ingress router then uses the destination PT field in a received packet and the forwarding state it established using BGP to choose the appropriate egress router within that same AS. It then writes the VC address for that egress router into the packet, *separately* from the end-host destination address in the packet. Thus, under RSR, this egress router address field is rewritten at each AS' ingress router.[4]

We now provide further details concerning RSR's workings.

2.0.0.3 Node Addressing.

There are three addressing entities in RSR, as introduced above. *Virtual Coordinates (VCs)* are in an n-dimensional space.[5] VCs reflect intra-domain TE constraints. They are *not* globally unique, but are rather AS-scoped. *Geographic Coordinates (GCs)* are in two-dimensional space, and are merely the physical positions of routers (if we are concerned about giving away the GC's of routers, we can use "synthetic" GC's using techniques in [15]). GCs *are* globally unique; a GC's low-order bits are a unique ID of a router or end host (*e.g.,* an Ethernet MAC address). Finally, a *Policy Tag (PT)* identifies a host or router's AS; it may be thought of as today's AS number.

In RSR, the DNS resolves a host-name to a GC address, and GC addresses are used in transport protocol control blocks.

A packet carries fields as follows:

Field	Description
src GC	source host geo coords[6]
dst GC	dst host geo coords
dst VC	dst host virt coords
dst PT	dst host policy tag[7]
egress GC	AS egress router geo coords
egress VC	AS egress router virt coords

We assume that each host and router knows its own position, and thus its GC address, and also knows its own AS, and thus knows its own PT. A sending host may thus fill in the src GC field for a packet it generates, and after a DNS lookup, may fill in the dst GC field as well. What remains to be discussed is how a destination's VC field and PT field are generated, and how they are written into packets.

2.0.0.4 Generating VCs.

As previously described, instead of performing a Dijkstra computation on the link state database, each RSR router *computes an embedding of the link-weighted topology onto an n-dimensional coordinate space.* The embedding algorithm we use is adapted from from Bourgain's algorithm [14]. This algorithm belongs to a body of literature dealing with embedding metric spaces into other metric spaces (see [9, 14] for an exposition of such techniques).

Given a weighted graph, we observe that we can construct a metric space on this graph by forming the full Dijkstra matrix D. Bourgain's algorithm shows how to embed (*i.e.,* calculate the VCs) a given set of n points forming a metric space (this metric space is D in our case) into $O(log(n))$ dimensions with $O(log(n))$ distortion. It is a simple probabilistic algorithm; because of space constraints, we do not cover its details here.

The resulting assignment of VCs has the property that the Euclidean distance between any two routers in this n-D coordinate system is "close" to the least path cost between those routers in the underlying topology. Thus, RSR approximately satisfies an ISP's TE requirements by computing an embedding with low distortion. Note that a router's VC can change under dynamics such as link weight changes and link failure/restores. These dynamics mimic changes in forwarding tables that occur in conventional routing.

We finally note that in RSR, the VC of a host is the VC assigned to its first-hop router; this VC can be discovered using standard router discovery mechanisms [4]. We defer discussion of how a sending host learns a destination host's VC and PT (which are volatile, and not available in DNS) until the end of this section.

2.0.0.5 Intra-Domain Routing.

An RSR router's forwarding table contains only the node addresses and the current virtual coordinates of its immediate neighbors.[8] As previously described, routers forward packets to neighbors that are progressively closer to the destination in VC space. This greedy forwarding will often, but not always, succeed. When no closer neighbor is available, *RSR falls back to using geographic face routing on GCs,* using the dst GC address field of the packet. Simply put, geographic face routing traverses faces of an almost [11] planar graph generated by CLDP.[9] Face routing continues until the packet reaches a router whose VC is closer to the destination than the router that couldn't forward greedily using the packet's dst VC address field.

Note that face routing can greatly increase the path stretch, since geographic routing is unaware of link weights. In fact, in our simulations (Section 3), we show this effect is significant. Fortunately, a simple *one-hop lookahead* heuris-

[4]This rewriting of addresses at AS boundaries is reminiscent of IPNL [8], but for different aims (small routing state, rather than flexibility), and by different means (using state obtained from BGP, rather than state obtained from DNS).

[5]In this work, $n = 16$.

[6]In the first packet of a connection, the source would include its own VC and PT in the options header so that the destination need not do discover them separately.

[7]In our embedding, dst VC and dst PT together occupy only 128 bits.

[8]We describe below a slight change to this that requires routers to maintain a little bit more information.

[9]In this paper, for reasons of space, we have omitted a description of how CLDP works. For our purposes, what is important here is that CLDP ensures failure-free geographic routing on arbitrary graphs.

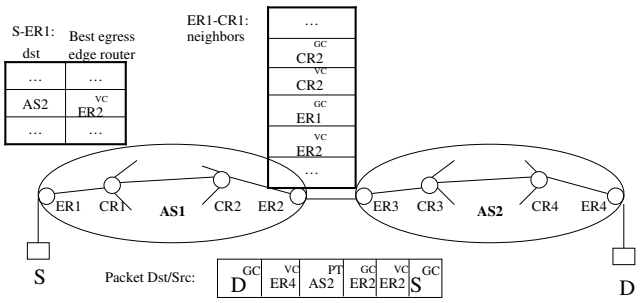

Figure 1: RSR example

tic works well to dramatically reduce stretch, as also shown in Section 3. In this technique, routers keep the VCs and GCs not only of their neighbors, but also of their neighbors' neighbors as well. This set of nodes is used both in the greedy VC routing phase, as well as in determining when to exit GC routing.

Finally, we emphasize that RSR only approximately satisfies ISP TE requirements, since not all packets take least-cost paths. Thus, *stretch* (the ratio of the cost of the actual path taken to the least cost path) is an important measure of RSR performance, and one we evaluate in Section 3.

2.0.0.6 *Inter-Domain Routing.*

To complete our description of RSR, we now outline how RSR implements policy routing. In RSR, ISPs (or autonomous systems) run BGP essentially unchanged, except for one crucial difference: instead of distributing IP prefix information in BGP, RSR disseminates opaque *policy tags*. A policy tag is an identifier associated with a set of destinations. RSR applies the same policy for routing to all destinations associated with a policy tag. The idea of policy tags is not new; policy atoms [2] are analogous to policy tags. A simple example of a policy tag is an AS number; all destinations within the AS are subject to the same policy.

When forwarding a packet to an arbitrary destination, an AS's ingress router extracts the dst PT field from the packet, then determines the VC of the egress router for that destination within the same AS, and writes that VC into the packet's egress VC field. Subsequently, within that AS, *the packet is routed using the egress VC field.* This is key to RSR's scaling, since core routers now need only maintain reduced routing state. Ingress routers, however, maintain state comparable to today's Internet. Unlike in today's routing system, where all intra-domain routers must run I-BGP and maintain full forwarding tables, the PT-to-egress-VC mapping need only be maintained at edge routers. Moreover, because mapping a packet's PT to its egress router requires a perfect-match and not a longest-prefix-match lookup, fast-path forwarding is significantly simplified at edge routers.

2.0.0.7 *Putting It All Together.*

We now pictorially describe how these various components work together in RSR. Figure 1 shows what forwarding state is kept in RSR to support end-to-end routing between hosts S and D, and how addresses are used inside that flow's packets.

A packet holds the GC addresses of the source and destination (S^{GC} and D^{GC}, respectively, in the figure). It also holds the VC and PT values of the destination ($ER4^{VC}$ and $AS2^{PT}$ respectively, in the figure). Finally, the egress VC value held is overwritten at each AS's ingress router (to be $ER2^{VC}$ in $AS1$, and $ER4^{VC}$ in $AS2$ in the figure).

How does the source learn $ER4^{VC}$ and $AS2^{PT}$ in the first place, given that a VC is ephemeral and not globally unique, and therefore unsuitable for keeping in DNS? The *first* packet from the source to the destination is routed geographically, and the destination's VC and PT is returned in response to that first packet (*e.g.,* piggybacked on an ACK or a transport-level handshake message). Subsequently, all packets from the source contain a (possibly dynamically changing) VC and PT of the destination. If the destination VC changes dynamically, as during long-lived connections, fast path routing on a stale destination VC would take the packet to the wrong last-hop router that discovers this problem (*e.g.,* D^{GC} falls outside the geographic region that the router handles), and returns an ICMP message with S^{GC} as the destination address so that the source can (re)discover the new destination VC. Likewise, when destination changes providers, sources discover new PT's.

The forwarding table that an edge router like $ER1$ stores in the line card of the S–$ER1$ link is shown in Figure 1. When a packet with a valid destination VC address enters $ER1$, it looks up the best egress edge router using the packet's destination PT field, and writes the resulting VC into the packet header ($ER2^{VC}$ in this case). Now that there is sufficient information inside the packet about the intermediate destination ($ER2$), $ER1$ uses the egress VC address ($ER2^{VC}$) as the destination address, and invokes the RSR forwarding algorithm to forward it to a core router $CR1$. Thus, edge routers do both a lookup to decide the egress, and a next-hop lookup based on the egress VC.

Core routers are simpler since, as described above, they can keep reduced state and forward on VCs alone. The forwarding state that a core router like $CR1$ keeps in its line card for the link $ER1$–$CR1$ is also shown in Figure 1. $CR1$ selects $CR2$ on this basis, and this forwarding process repeats until the packet reaches $ER2$.

3. RESULTS

In this section we present the performance of the core components of RSR (virtual co-ordinate assignment, and forwarding strategy as described in Section 2) by simulating them on realistic ISP topologies. Our primary goal is to demonstrate the feasibility of RSR on realistic topologies; it is not immediately obvious that our embedding strategy or the use of face routing will result in reasonable path stretches. We developed a simulator in C that, given a topology with associated link weights, computes the embedding of the topol-

ogy in higher dimensions, and forwards packets between any pair of nodes.

We evaluate RSR on ISP topologies inferred by Rocketfuel [16]. These topologies are annotated with the geographic location of routers, and inferred link weights. We use this information to compute the router VCs, and to implement face routing.

At first glance, it appears that RSR would perform poorly since it employs an embedding that can distort paths, and it uses geographic face routing which is oblivious to link weights. Our primary measure for validating RSR is *path stretch*: the ratio of the total weight along a path taken by RSR to the least cost path.[10] Thus, one metric we use for measuring RSR performance is the *average path stretch* taken over all pairwise paths in the topology.

However, average stretch alone does not accurately capture the performance of RSR. If paths A and B have the same stretch, but B is significantly longer than A, then B should arguably count for more when measuring performance since packets traversed using B consume more resources. We account for this using two different metrics. First, we compute a metric called *aggregate stretch* which is the ratio of the sum of all pairwise RSR paths to the sum of all pairwise least cost paths. This aggregation naturally accounts for long paths. Second, we compute a metric called *discounted stretch*, which is defined as the average over all paths of *discounted path stretch*. This latter quantity is defined for a given path as $\frac{C+p_a}{C+p_s}$, where C is a constant, p_a is cost of the path chosen by RSR, and p_s is the least cost path between the same endpoints. Intuitively, discounted path stretch discounts paths of cost less than C in computing stretch.

Table 1 displays the performance of RSR on a variety of ISP topologies. For each of our simulations, we use an embedding in 16 dimensions and assign 5 bits per dimensions. For context, Table 1 also shows the results for grid and random topologies. Random topologies are random graphs in the unit square with randomly assigned link weights chosen uniformly between 1 and 10 (this range is comparable to ISP link weight range). Grid topologies are $n \times n$ grids whose link weights are similarly assigned.

Table 1 shows that, for most of our ISP topologies, stretch values are less than two. Indeed, for some of the smaller topologies like Exodus, RSR achieves stretches close to 1.1. This is reassuring, since these values are comparable to those reported for the stretch induced by policy routing that counts path hops [18]. At the same time, this result is surprising since it isn't *a priori* clear that RSR could achieve such low stretch values. We explain this as follows. Stretch has two components: one induced by greedy routing on the VCs, and the other induced by face routing (both contributions are shown in Table 1. We found that the former component inflates stretch by only a small extent (not much more than $\sim 2X$) no matter how much % of the path is spent in greedy mode, while the stretch due to face routing is high (50 or higher) if we don't use one level lookahead (which causes a comparatively higher % of the path to be spent in face routing mode). Since face routing is inherently oblivious to link weights, the impact on the overall stretch is severe. Our strategy of using a 1 hop lookahead both during greedy routing and checking when we can exit face routing markedly reduces the impact of face routing by causing the packet to spend a large percentage of its path in greedy mode, and enter face routing for only short segments. Consequently, each face routing segment incurs a smaller stretch that is comparable to greedy.

In addition, our simulations incorporate another simple trick that reduces the cost of face walks. We observed that our ISP topologies have many degree 1 nodes. These are basically leaf nodes from which there is no where to go except back during face routing. We avoid using such links during face routing.

Two observations about the table are worth highlighting. Discounted stretch is generally lower than average stretch, indicating that a number of paths which contribute to the increased stretch are short paths. This is particularly evident for the Sprint topology, whose average stretch is slightly higher than 2, but whose discounted stretch is comparatively low. In addition, aggregate stretch is sometimes higher than average stretch and sometimes lower, indicating that some topologies have a greater fraction of longer paths with greater stretches than others. From the Table, it can be seen that this variation is not much.

Aside from stretch, there are several secondary performance questions in RSR. To these we present some preliminary answers. One such question is: what address size is sufficient, and how many dimensions are sufficient for the embedding? We extensively examined stretch performance for various values of these quantities, and found that an 80-bit address on a 16-dimensional embedding performs quite well (performance saturates for higher address bit budgets and higher number of dimensions). Also, given a fixed address budget, we see a threshold for the bit resolution/dimension below which stretch degrades. Thus, it seems to always be preferable to use a lower (*e.g.* 16) dimensional embedding with resolution higher than the threshold (5 bits in our case), than a higher dimensional embedding with lower bits per dimension (say 20D with 4 bits/D or 80D with 1bit/D). These observations are specific to our choice of embedding algorithm (Bourgain's embedding), but we expect to see such "sweet spots" for other other algorithms as well.

Given that RSR uses a one-hop lookahead for greedy routing, another question is: what is the precise amount of state that core routers need to keep? As Table 1 shows, both average and maximum states for all our ISP topologies are relatively low; Sprint incurs the highest maximum state. In general, one would expect core routers to have relatively low state despite the existence of power-laws in router topology; that is because most of the high-degree nodes in the router

[10] In this work, lacking ISP data, we do not quantify the corresponding latency change (most likely to be higher for most paths).

Topology	Total Nodes	Average Degree	Maximum State	Average State	Average Stretch	Aggregate Stretch	Discounted Stretch (C=3)	Greedy Stretch	Face Stretch	Embedding Time /Dijkstra Time
Grid	400	3.800	12	11.008	1.439	1.497	1.413	1.763	1.124	0.536
Random	400	18.51	342	253.685	2.287	2.252	1.691	1.695	2.205	0.575
AS 1239 Sprint	315	5.171	135	37.810	2.153	1.900	1.724	1.723	5.741	0.624
AS 1755 Ebone	86	2.744	28	11.826	2.167	2.348	1.966	1.627	3.625	2.72
AS 3257 Tiscali	181	3.154	71	21.613	1.77	1.707	1.592	1.744	1.91	1.53
AS 3967 Exodus	78	2.769	32	13.308	1.204	1.194	1.170	1.137	1.377	3.797
AS 6461 Abovenet	141	4.319	71	22.475	1.583	1.650	1.472	1.526	1.623	1.56

Table 1: Results for a 16 dimensional embedding and 80 bit address

topology tend to be at the edges [13]. Also, note that the random topology has a much higher maximum state. This explains its surprisingly low stretch given that node locations are completely random, and that the average node degree is about $4X$ more than that of the largest ISP (Sprint); by keeping so much lookahead state, RSR is able to more frequently avoid face routing on the random graph.

Finally, the cost of computing the embedding is an important performance issue. Table 1 shows that the the time taken to compute the embedding is comparable to, or slightly larger than, the time taken to compute least cost routes using Dijkstra's algorithm.

In summary, our results indicate that RSR performs reasonably well by several metrics. The results we present are obtained without much performance tuning, save for a couple of optimizations we discussed already. In general, we expect that we can improve the overall performance of RSR by more carefully examining the various design choices and their attendant trade-offs. We leave the evaluation of RSR's end-to-end path stretch performance across multiple ISPs to future work.

4. CONCLUSION

We have presented RSR, a novel and feasible approach to reducing forwarding state size at core routers, that allows ISPs to engineer traffic and control routing policy using the same IGP link weight and BGP policy mechanisms they use today. RSR brings the scalability of geographic routing to the Internet by embedding link weights into a virtual coordinate system, and routing greedily in that virtual coordinate system; and by falling back on greedy routing on physical node locations to ensure reachability in all network graphs. Thereby, RSR drastically reduces the state requirement at an ISP backbone's core routers, which only need keep state proportional to their fanout. Our simulation results demonstrate that on real ISP backbone topologies, RSR achieves this state reduction with bearable path stretches. While we demonstrated in this paper that RSR can meet TE rules represented as IGP link weights, it would be interesting in future to examine how to make VC routing work across more complex TE goals embodied by richer MPLS designs.

5. REFERENCES

[1] Bose and Morin. Online routing in triangulations. In *ISAAC: 10th International Symposium on Algorithms and Computation*, 1999.

[2] A. Broido and K. C. Claffy. Analysis of routeviews bgp data: Policy atoms. In *Proc. of Network-related Data Management Workshop*, May 2001.

[3] S. Deering. Metro-based addressing.

[4] S. Deering. ICMP router discovery messages (rfc 1256), September 1991.

[5] M. Degermark, A. Brodnik, S. Carlson, and S. Pink. Small Forwarding Tables for Fast Routing Lookups. *SIGCOMM*, 1997.

[6] W. Eatherton, Z. Dittia, and G. Varghese. Tree bitmap: Hardware/software ip lookups with incremental updates. *ACM SIGCOMM Computer Communications Review*, 34(2), 2004.

[7] G. G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, Information Sciences Institute, Mar. 1987.

[8] P. Francis and R. Gummadi. Ipnl: A nat-extended internet architecture. In *Proc. of the ACM SIGCOMM Conference on Network Architectures and Protocols '01*, 2001.

[9] P. Indyk. Algorithmic applications of low-distortion embeddings. In *Proc. of 42nd IEEE Symposium of Foundations of Computer Science '01*, 2001.

[10] B. Karp and H. T. Kung. Gpsr: Greedy perimeter state routing for wireless networks. In *Proc. of 6th ACM Int. Conf. on Mobile Computing and Networking (MobiCom '00)*, 2000.

[11] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker. Practical and robust geographic routing in wireless networks. Technical Report 04-832, Department of Computer Science, University of Southern California, 2004.

[12] F. Kuhn, R. Wattenhofer, and Z. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC-03)*, 2003.

[13] L. Li, D. Alderson, W. Willinger, and J. Doyle. A first principles approach to understanding the internet's router-level topology. In *Proc. of the ACM SIGCOMM Conference on Network Architectures and Protocols '04*, August 2004.

[14] J. Matousek. *Lectures on Discrete Geometry*. Springer, 2002.

[15] A. Rao, C. Papadimitriou, S. Shenker, and I Stoica. Geographic routing without location information. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, 2003.

[16] N. Spring, R. Mahajan, and D. J. Wetherall. Measuring isp topologies with rocketfuel. In *Proc. of the ACM SIGCOMM Conference on Network Architectures and Protocols '02*, 2002.

[17] V. Srinivasan and G. Varghese. Fast address lookups using controlled prefix expansion. *ACM Transactions on Computer Systems*, 17(1):1–40, Feb. 1999.

[18] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The impact of routing policy on internet paths. In *INFOCOM*, 2001.