On the Scalability of Routing With Policies

András Gulyás, Gábor Rétvári, *Member, IEEE*, Zalán Heszberger, *Member, IEEE*, and Rachit Agarwal

***Abstract*—Today's ever-growing networks call for routing schemes with sound theoretical scalability guarantees. In this context, a routing scheme is scalable if the amount of memory needed to implement it grows significantly slower than the network size. Unfortunately, theoretical scalability characterizations only exist for shortest path routing, but for general policy routing that current and future networks increasingly rely on, very little understanding is available. In this paper, we attempt to fill this gap. We define a general framework for policy routing, and we study the theoretical scaling properties of three fundamental policy models within this framework. Our most important contributions are the finding that, contrary to shortest path routing, there exist policies that inherently scale well, and a separation between the class of policies that admit compact routing tables and those that do not. Finally, we ask to what extent memory size can be decreased by allowing paths to contain a certain bounded number of policy violations and, surprisingly, we conclude that most unscalable policies remain unscalable under the relaxed model as well.**

***Index Terms*—Compact routing, policy routing, routing algebras, routing scalability.**

I. Introduction

ROUTING scalability is one of the most fundamental goals of any network architecture. As the networks grow in size, it is highly desirable to use routing schemes that do not impose stringent storage requirements at nodes constituting the routing infrastructure. The area of compact routing investigates the theoretical scalability properties of routing within a formal framework. Within this framework, the fundamental memory–stretch[1] tradeoff is by now well understood—if small routing tables are desired then one must give up on the path length.

With the research efforts in compact routing moving from theoretical front to practical implications [1], [2], we ask ourselves if theoretical results are ready yet for applications in practice. We notice a fundamental obstruction—*all* compact routing studies thus far make a fundamental assumption of each node desiring the shortest path to the destination. While in reality, with perhaps the exception of sensor networks [3], networks that make the scalability problem nontrivial are precisely the ones where the desired paths are not necessarily the shortest ones.

Indeed, the example to start with is the Internet. Each router in the Internet stores, for each destination prefix, a next-hop entry that is a result of a complicated decision process integrated within the internal and external Border Gateway Protocol. While this decision process encompasses path length at a low level, what predominantly governs path selection are the individual business interests of service providers, the router's distance to the border routers, and other local preferences that may have little to do with path length. Another intriguing case for *compact routing with policies* is that of content-centric routing—an extensive research effort in which users route to contents, rather than the hosts serving the content [4], [5]. In this context, the scalability problem is indeed at the forefront—the number of destinations toward which routing information must be stored at a node is not just in the order of the hundreds of millions of other nodes, but rather the trillions of contents that constitute the World Wide Web. Routing scalability while incorporating policies is a nontrivial challenge within the framework.

This paper is the result of our quest for a technique to integrate policies within the framework of compact routing. However, we are not solely motivated by filling the gaps; the real driving force is our desire to understand if the tradeoff between memory and path cost is indeed a fundamental one. That is, we ask ourselves the following: 1) Can we achieve policy-compliant routing in large networks with compact routing tables? 2) If not, can we relax the policy constraints at nodes, just as traditional compact routing relaxes the shortest path requirement, to achieve compact routing tables?

We do acknowledge that it would be too optimistic to expect a positive answer to both of the above questions in the most general framework. Policy-induced routing schemes are fundamentally hard to model [6], [7], have a complicated combinatorial structure [8] and, unlike shortest path routing, may not even result in each node having a path to the destination [9]. We also realize that with network architectures like content-centric, it is even hard to predict what style of policies will be implemented at the time these architectures are realized in practice. All we know is that the policies that are predominantly used in today's networks are induced by economic relationships, and there are no reasons to believe that this is going to change arbitrarily in the near future. Keeping this in mind, we lay down some minimalistic assumptions on the routing policies we consider; these

Manuscript received January 15, 2013; revised February 05, 2014; accepted June 30, 2014; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J. Wang. This work was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013).Also the research was partially supported by the Hungarian Scientific Research Fund (grant No. OTKA 108947).

A. Gulyás and Z. Heszberger are with the MTA-BME Future Internet Research Group, Budapest University of Technology and Economics, Budapest 1117, Hungary, and also with the Information System Research Group, Hungarian Academy of Sciences (MTA), Budapest 1245, Hungary (e-mail: gulyas@tmit.bme.hu; heszberger@tmit.bme.hu).

G. Rétvári, is with the MTA-BME Future Internet Research Group, Budapest University of Technology and Economics, Budapest 1117, Hungary (retvari@tmit.bme.hu).

R. Agarwal is with the University of California, Berkeley, Berkeley, CA 94720 USA (e-mail: ragarwal@berkeley.edu).

Digital Object Identifier 10.1109/TNET.2014.2345839

[1]Stretch is defined as the maximum ratio of the length of the path used by a routing scheme to the actual shortest path.

assumptions are widely believed to hold in today's network and do provide us with an opportunity to attack the problem within a formal framework.

Our main contribution is a *separation* between the class of policies that can be implemented with compact routing tables and those that are not. More specifically, we show that when the set of available paths is filtered using the class of policies that are used in today's networks, there exists an *incompressibility* point—a policy before which compactness in routing tables is admissible and beyond which it is not. This is rather surprising, as traditional compact routing establishes that it is essential to relax the shortest path routing requirement to achieve compact routing tables; our results, on the other hand, suggest that there exists a class of routing policies that admit compact routing tables without relaxing the path selection criterion in any ways. This partially answers the first question posed above.

For the class of policies that do not admit compact routing tables, we present another surprising result. We show that there exist policies that do not admit compact routing tables, even if one is willing to violate policies at a constant number of nodes along the path in the hope of reducing memory requirements. This is in sharp contrast to traditional compact routing results, where one can trade off the path length to achieve compact routing tables.

In summary, this paper marks a step in the direction of moving beyond the memory–stretch tradeoff in networks where paths are selected based on routing policies, rather than simply the path length. On the positive front, we gain hope of designing policies where each node in the network may achieve its most desired path without storing a large number of routing entries; on the other hand, it highlights some of the policies that should be avoided within future network architectures, if compact routing tables were a goal.

The rest of this paper is organized as follows. After reviewing related work (Section II), we characterize the policy classes that fundamentally arise in today's networks (Section III). We formalize these policies rigorously (Section IV), and then we state our main result: a separation between scalable and unscalable routing policies (Section V). Finally, we study to what extent memory size can be reduced by relaxing policy compliance (Section VI), we draw the conclusions, and we sketch some future research directions (Section VII).

II. Related Work

Our work uses and provides contribution to two key areas of related work.

Traditional Compact Routing: Scalability of shortest path routing gained much attention after Gavoille *et al.* gave strong lower bounds on the memory requirements needed to implement it [10], [11]. In particular, they proved that it is impossible to achieve routing tables that are sublinear in the size of the network, if one insists on routing along shortest paths. Even if the shortest path requirement is relaxed, allowing for paths with (integer) stretch $2k-1$, the memory requirements cannot scale better than $\Omega(kn^{1+1/k})$ [12]. Subject to a conjecture of Erdős, this lower bound is also tight—there exist routing schemes that admit this point within the tradeoff space for any integer k [12]. For the particular case of stretch 3, [12] and [13] give compact routing schemes where local storage increases only with the square root of the network size.

There is very little hope of these techniques being helpful in our case; indeed, nodes selecting paths based on policies changes the three underlying assumptions made explicitly in previous works. First, it is hard to model a policy-induced graph by either directed or undirected graphs. Second, if paths are selected based on policies, connectivity provided by the underlying physical network topology does not imply reachability—two nodes that are physically connected may not be able to reach each other due to the absence of a policy-compliant path. Finally, the *cost* of a path is not necessarily the sum of the weights of the links on the path, and hence, defining a cost function on available paths in order to meet the fundamental requirement of costs constituting a metric space becomes a nontrivial task.

Scalability of Routing With Policies: One fundamental obstruction in formalizing the scalability problem with routing policies is the lack of a formal model that captures a large class of routing policies [6]–[8]. Independent administrative domains implement routing policies at both the border routers (mostly based on economic relationships with neighboring domains [14]–[16]) and at internal routers (based on internal routing protocol, traffic engineering, distance to border routers, etc. [15]). Having hundreds of millions of routers implementing routing policies naturally raises scalability concerns; the proposals on content-centric networking with routers implementing policies per content [17] will only aggravate the problem.

Indeed, incorporating policies within the compact routing framework has been known for a while to be an open problem [18], [19]. However, the lack of a formal scalability study is perhaps not too surprising precisely due to our inability to model policies within a formal framework. Very recently, an algebraic framework was proposed as a possible tool to attack the problem [20]; it is essentially this framework that we exploit to derive our results.

III. Routing Policies

The lack of a unified model has led to study routing policies within a coarse grained classification; it is often feasible to understand the behavior of each class while ignoring interaction of policies from separate classes [15], [16]. In this section, we outline three such classes that fundamentally arise in today's networks. While possibilities exist to extend our model to a more general class of policies, these three classes are sufficient to derive the separation result we set our focus on—identifying the policy before which compactness in routing tables is admissible and beyond which it is not.

To make discussion in this section succinct, we use the notion of *actors* to identify different levels of granularity at which routing policies may be implemented. For instance, actors may refer to autonomous domains when we discuss policy issues at border routers or may refer to routers themselves when we discuss policy issues at internal routers. In essence, the term actor refers to a basic element in the network that can have routing policies independent of the policies of other actors in the network.

It is widely believed that at the granularity of independently administered domains, routing policies are a result of bilateral economic relationships between neighboring actors, taking the form of either a *provider–customer* or a *peering* relationship [14]–[16], [21], [22]. In the former, traffic between the two domains is paid for by the customer, while in the latter, traffic exchange is cost-free subject to traffic flow constraints. Within a single domain, routing policies at actors take multiple factors into account including distance to border routers, external and internal routing advertisements, traffic engineering, etc. In this paper, we concentrate on a well-defined subset of these policies. Our aim is to come up with models general enough to be practically relevant while, at the same time, also sufficiently simple to admit a comprehensive mathematical analysis.

Valley-Free Routing: The first routing policy that we consider is valley-free routing [16], [21]. This policy encompasses the basic economic fact that the flow of traffic must obey the flow of cash. The policy dictates that an actor A can use a link to a neighboring actor B to route traffic if and only if either the incoming traffic is from a customer or B is a customer of A. All paths that induce valley-free routing have the same preference; the only concern this policy addresses is not violating elemental business rules.

Local Preferences: The second routing policy we consider is the most general one within our framework—actors decide on the next-hop based on their own preferences, which we assume to be motivated by individual economic interests [14]–[16]. For instance, an actor might prefer to route traffic via a customer as this comes without expenditures and, in absence of such a path, it is indifferent to whether the next hop is a provider or a peer.

Path Length: Finally, we consider routing policies where actors select paths based on their lengths. While this policy has been thoroughly examined in traditional compact routing literature, we choose this for two reasons: First, when combined with the earlier two policies (that is, let paths be filtered through multiple policy criteria as in today's networks), it helps us establish some surprising results; second, path length still forms a part of the path selection process in today's networks [15].

Potpourri: We consider three combinations of the above policies as the ones relevant in practice. In particular, we analyze the scalability of pure valley-free routing, valley-free routing with local preferences, plus the previous two with minimizing path length. Indeed, this latter policy most resembles the actual decision process used in today's networks.

IV. Notations and Formal Model

We model the network as a connected, simple graph $G(V, A)$, with each node in the graph denoting an actor and an arc denoting a physical connection between the two actors. We follow the standard notation to assign directions to the arcs [23]: The arrowheads mark the direction of the cash flow (see Fig. 1). At connections with no cash flow (peering links, or internal routers for instance), we either use a bidirectional arc, or ignore the direction for ease of discussion. For instance, in Fig. 1, c_1 is a customer of p_1, p_2 and p_3 are peers, and t_1 is a provider of both p_1 and p_2.

Throughout the paper, we denote a path from node s to node t using the notation p_{st}. A routing policy is represented as a

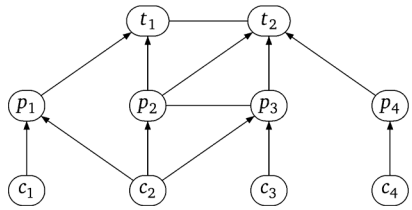

Fig. 1. Visual representation of a network with policies.

function $\mathrm{Pol}(\cdot) : \mathcal{P}_{st} \rightarrow P^*_{st}$, which operates over a set of *available* paths from node s to node t and selects a *preferred* path P^*_{st} based on predefined rules or policies at node s. To describe these rules concisely, we adopt the notion of routing algebras from the literature [22], [24], [25]. Routing algebras provide us with a unified tool to capture the inherently bilateral business agreements between actors using abstract weights (described above), and define policy-compliant paths in terms of these weights. The rest of this section briefly introduces the routing algebra and defines the algebra for the routing policies discussed in last section.

A routing algebra $\mathcal{A}$ is defined as a totally ordered semigroup of abstract link weights, with a compatible infinity element [20]. Formally, a routing algebra $\mathcal{A}$ is defined by the tuple $(W, \phi, \oplus, \preceq)$, where W is the set of abstract weights that can be assigned to arcs, $\phi \notin W$ is a special infinity weight meaning that an arc/path is not traversable (that is, not policy-compliant), $\oplus$ is a right-associative[2, 3] composition operator for weights, and $\preceq$ is weight comparison expressing a preference on the paths/arcs.

We now discuss how routing algebras allow us to succinctly capture the routing policies. Given a path $P = (v_1, v_2, \ldots, v_k)$ in the network, the weight $w(P)$ of P is obtained by combining the weight of arcs on the path

$$w(p) = \bigoplus_{i=1}^{k-1} w(v_i, v_{i+1}).$$

Using such a notion of path weight simplifies defining policies; for instance, a path that is *preferred* based on the policies can be captured within the algebra $\mathcal{A}$ as the path with the smallest weight according to $\preceq$

$$\mathrm{Pol}(\mathcal{P}_{st}) = P^* : w(P^*) \preceq w(P), \forall P \in \mathcal{P}_{st}.$$

Within the algebraic framework, one can capture a fairly general class of policies. To start with an example, the algebra for min-hop routing is defined as

$$\mathcal{S} = (\{1\}, \infty, +, \leq). \tag{1}$$

For shortest path routing over general positive weights, we have the following algebra: $(\mathbb{R}^+, \infty, +, \leq)$. For more details on routing algebras, refer to [22], [24], and [25]. In what follows, we define the algebras for the routing policies described in Section III.

[2]Right-associativity means that weights compose from the right, i.e., $w_1 \oplus w_2 \oplus w_3 = w_1 \oplus (w_2 \oplus w_3)$.

[3]This is not crucial for the model, but it will bring our generic framework closer to existing practice [22].

TABLE I
WEIGHT COMPOSITION IN VALLEY-FREE ROUTING

$\oplus$	c	r	p
c	c	ϕ	ϕ
r	r	ϕ	ϕ
p	p	p	p

A. Valley-Free Routing

The formal algebraic description of valley-free routing is as follows [22]. We assign *abstract* weights to links in the network from the set $\{c, r, p\}$. An arc directed from a customer to a provider is assigned weight p, and one from a provider to the customer is assigned weight c. In case of peering relationships, we assign weight r in both directions. Then, a valley-free path is one that traverses at most one link of weight r, does not cross any p links after an r or a c link, and does not use an r link after a c link. For instance, in Fig. 1, path $c_1 \rightarrow p_1 \rightarrow t_1 \rightarrow t_2 \rightarrow p_3$ is valley-free, but $p_2 \rightarrow p_3 \rightarrow t_2$ and $p_2 \rightarrow c_2 \rightarrow p_3$ are not. Consider the following definition.

Definition 1: Valley-free routing is defined as the algebra $\mathcal{A}_1 = (W, \phi, \oplus, \preceq)$, where $W = \{c, p, r\}; \oplus$ is as in Table I; and $c = r = p \prec \phi$.

In the above algebra, the abstract weight of a path consisting of two provider links, for instance, is $p \oplus p = p$, while $r \oplus r = \phi$ states that the succession of two peer links in a path is prohibited. The term $c = r = p$ means that the relation $\preceq$ orders the same preference to whether the next hop on the path is a customer, provider, or peer as long as the path is a valley-free path.

B. Valley-Free Routing With Local Preference

Next, we describe the routing algebra for policies that induce *local preferences* over the set of available valley-free paths. Local preferences are the set of rules that allow actors to exercise their own business interests, operational concerns, and other individual criteria in selecting paths. Among an extremely large number of possible local preference rules, perhaps the one that is widely believed to be most applicable is actors preferring customer paths over provider and peer paths. Consequently, we restrict ourselves to this elementary local preference rule in the sequel.

This policy can be captured within the algebraic framework by a simple modification to the preference operator in the algebra for valley-free routing [22], [25].

Definition 2: Valley-free routing with local preference is defined as the algebra $\mathcal{A}_2 = (W, \phi, \oplus, \preceq)$, where $W = \{c, p, r\}; \oplus$ is as in Table I, and $c \prec r = p \prec \phi$.

Note that the weight composition for this policy is unchanged when compared to the valley-free routing policy; only the preference operator has changed. In terms of our sample network in Fig. 1, the rule $c \prec p$ simply states that the path $p_1 \rightarrow c_2$ path of weight c is preferred over the path $p_1 \rightarrow t_1 \rightarrow p_2 \rightarrow c_2$ path of weight p.

C. Valley-Free Routing With Local Preference and Shortest Path

Finally, we consider the routing policy that selects, among all paths that are local preference-compliant, the one with minimum number of actors along the path. To formally define the routing algebra for this policy, we introduce a useful algebraic construct called *lexicographic product* [26].

Definition 3: Given two routing algebras, $\mathcal{A} = (W_{\mathcal{A}}, \phi_{\mathcal{A}}, \oplus_{\mathcal{A}}, \preceq_{\mathcal{A}})$ and $\mathcal{B} = (W_{\mathcal{B}}, \phi_{\mathcal{B}}, \oplus_{\mathcal{B}}, \preceq_{\mathcal{B}})$, the *lexicographic product* of $\mathcal{A}$ and $\mathcal{B}$ is a routing algebra $\mathcal{A} \times \mathcal{B} = (W, \phi, \oplus, \preceq)$, where we have the following.

- $W = W_{\mathcal{A}} \times W_{\mathcal{B}}, \phi = (\phi_{\mathcal{A}}, \phi_{\mathcal{B}})$.
- $(w_1, v_1) \oplus (w_2, v_2) = (w_1 \oplus_{\mathcal{A}} w_2, v_1 \oplus_{\mathcal{B}} v_2)$ for all $w_1, w_2 \in W_{\mathcal{A}}$ and $v_1, v_2 \in W_{\mathcal{B}}$.
- $(w_1, v_1) \preceq (w_2, v_2) = \begin{cases} v_1 \preceq_{\mathcal{B}} v_2, & \text{if } w_1 =_{\mathcal{A}} w_2 \\ w_1 \preceq_{\mathcal{A}} w_2, & \text{otherwise.} \end{cases}$

A lexicographic product essentially states that when two policies are combined into one, the first takes precedence and the second only acts as tie-breaker. Then, the routing policy that favors the path through the fewest hops among all valley-free paths with the same local preference is the lexicographic product of $\mathcal{A}_2$ and the min-hop routing algebra $\mathcal{S}$ (1).

Definition 4: Valley-free Routing with Local Preference and Shortest Path First is defined by the algebra $\mathcal{A}_3 = \mathcal{A}_2 \times \mathcal{S}$.

For instance, in Fig. 1, paths $p_2 \rightarrow t_2 \rightarrow p_3$ and $p_2 \rightarrow t_1 \rightarrow t_2 \rightarrow p_3$ are both valley-free and local preference-compliant, but the former is preferred due to it being shorter.

In line with the rest of the policy routing literature, we make two fundamental model assumptions [9], [14]–[16].

Assumption 1: There are no provider loops in the network.

Assumption 2: Global reachability: Each actor has a policy-compliant route to every other actor.

These assumptions will play a crucial role in our theoretical scalability analysis, as revealed in Section V.

V. SCALING PROPERTIES OF THE POLICY ROUTING MODELS

Next, we turn to the analysis of the theoretical scaling properties of policy routing models. To do that, first we have to put it more clearly what scalability actually means.

We adopt a generic model from [10] and [11]. We are given a network of n nodes, represented by the graph $G(V, A)$, and a routing policy, represented by the algebra $\mathcal{A}$, and the task is to find a worst-case upper bound on the memory requirements needed to implement the policy. We generally suppose that node degrees can be in the order of $O(n)$.

Suppose that there is a hypothetical routing function R_u implemented at each node $u \in V$, responsible for forwarding incoming packets to the right next-hop through the appropriate outgoing interface. In shortest path routing, for instance, the routing function must choose the next-hop along the shortest path. In general policy routing, however, it is the policy, and implicitly the underlying algebra $\mathcal{A}$ through the relation $\preceq$, that determines the right way to forward a packet. The model is quite liberal with what the routing function R_u is allowed to do with the packet: It can perform arbitrary calculations on it; it can swap the header or the contents in any way; it has access to the entire graph topology, etc.; the only requirements are that: 1) the emergent paths must be policy-compliant; and 2) the number of bits $M(R_u)$ needed to encode the routing function on node u is minimal. Additionally, we are free to assign addresses in the network, under the restriction that address size cannot exceed $c \log n$ bits for c constant.

Routing scalability is then defined in terms of the maximum storage space attainable by the best routing function $M(R_u)$ existing in the set of all routing functions $\mathcal{R}$ implementing the algebra $\mathcal{A}$, experienced over all nodes u and over all graphs $\mathcal{G}_n$ of size n [10], [11]

$$M_{\mathcal{A}} = \max_{G \in \mathcal{G}_n} \min_{R \in \mathcal{R}} \max_{u \in V} M_{\mathcal{A}}(R, u).$$

We say that a policy $\mathcal{A}$ is *incompressible* if there exists a graph construction in which some routers need $\Omega(n)$ bits of storage to implement the policy. This usually signifies that the routing policy does not scale well because even within our liberal model, there is a network in which at least one node needs memory whose size increases linearly with the size of the network. Otherwise, the policy is *compressible*.

For shortest path routing, the following negative result exists [10], [11].

Proposition 1: Min-hop routing is incompressible.

Similar compressibility analysis for a broad range of routing policies was given in [20]. For general policy routing, however, no such characterizations exist apart from some preliminary results appearing in [20]. We generalize the treatment from [20] to the three policy routing models introduced in Section IV.

A. Valley-Free Routing

In valley-free routing, the only requirement for a path to be policy-compliant is to contain no valleys. In terms of our algebraic description $\mathcal{A}_1$, we require that the weight $w(p)$ of a policy-compliant path p is such that $w(p) \prec \phi$. We ask whether there is a routing algorithm that can implement valley-free routing with sublinear storage space. We answer this question in the affirmative, under the principles of the generic routing framework introduced in Section III.

Theorem 1: Under the assumptions: 1) that no provider loops exist (Assumption 1); and 2) global reachability (Assumption 2), there exist a route distribution and addressing scheme that implements valley-free routing with $O(\log n)$ storage at each router.

Proof: First, we extract a special subgraph, called the valley-free frame, in which all paths are valley-free. Under our assumptions, the valley-free frame always exists. Then, we prove that routing along the valley-free frame is possible with $O(\log n)$ bits memory per router.

First, collect the nodes corresponding to actors that do not have a provider into a set $\mathcal{T}$. Each remaining node has at least one provider link emanating from it: Choose an arbitrary provider, and let this be the preferred provider of the node. Do this recursively until a node in $\mathcal{T}$ is hit, and finish when a preferred provider is assigned to every node not contained in $\mathcal{T}$. One easily checks the following.

1) In the absence of provider loops, the above process terminates and organizes the nodes into a forest.
2) There is exactly one node from $\mathcal{T}$ in each tree of the forest.
3) Each $v \notin \mathcal{T}$ is associated with strictly one tree; let the unique node in the tree belonging to $\mathcal{T}$ be the root node of v, denoted by $t(v)$.
4) Each $v \notin \mathcal{T}$ has exactly one link toward a provider.

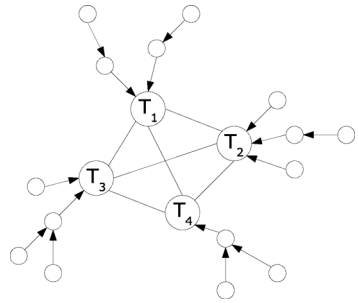

Fig. 2. Valley-free frame: a peering full-mesh of nodes in $\mathcal{T}$ plus a set of trees hanging on the root nodes in $\mathcal{T}$.

5) For any $v \notin \mathcal{T}$, the unique v to $t(v)$ path consists of provider links exclusively.
6) By global reachability, nodes in $\mathcal{T}$ are connected into a full-mesh of peer links.

We observe that all paths in the valley-free frame are indeed valley-free. An example is given in Fig. 2.

We assign addresses as follows. First, we label the nodes in $\mathcal{T}$ using at most $\log n$ bits. Next, for each $t \in \mathcal{T}$ to the nodes inside the tree hanging on t, we assign a $c \log n$-bit-size label using the tree-routing scheme in [27]. Then, for a node v, the address is constructed as a concatenation of the label of the root node of v plus the in-tree address (the root of a node $v \in \mathcal{T}$ is v itself)

$$\underbrace{\text{label assigned to } t(v)}_{\log n \text{ bits}} \; \underbrace{\text{in-tree label of } v}_{c \log n \text{ bits}}.$$

Upon forwarding a packet, a node first examines the initial $\log n$ bits of the destination address. If it matches the label of $t(v)$, v uses tree routing [27] to forward the packet inside its own tree. Otherwise, v forwards the packet toward $t(v)$. What remains to be done is to ensure that nodes inside $\mathcal{T}$ can forward packets to any other node in $\mathcal{T}$. This is done by labeling the outgoing links of $v \in \mathcal{T}$ with the label of the in-$\mathcal{T}$ node to which they are attached and forward a packet on the link whose label equals the first $\log n$ bits of the destination address. Pure tree routing requires $O(\log n)$ bits of storage on each node [27], to which our scheme adds only $\log n$ bits. Therefore, valley-free routing is compressible. ■

B. Valley-Free Routing With Local Preference

Next, we extend valley-free routing with the local preference rule that customer paths are preferred over any other types of paths. In terms of the algebraic representation $\mathcal{A}_2$, we now have the preference rule $c \prec r \preceq p$. We show that even this minimal local preference configuration, if set at each node, makes policy routing incompressible. To prove this result, we trace back this policy to shortest path routing. Then, the Proposition 1 on the incompressibility of shortest path routing will establish the required result.

Definition 5: Let G be a graph, and let C and T be two distinct sets of nodes of G. C is called the set of *constrained nodes*, and T is the set of *target nodes*. A *shortest path preserving labeling* is a labeling of the links of G, so that for any $u \in C$ and $v \in T$, a $u \to v$ path is preferred according to the actual routing policy if and only if it is a min-hop path in G.

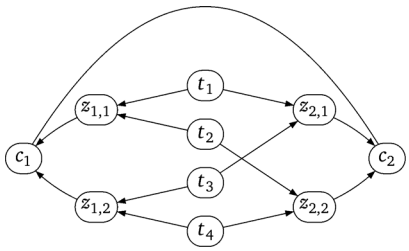

Fig. 3. Sample graph for $p = 2, \delta = 2$ if the words for the target nodes are [1, 1], [1, 2], [2, 1], and [2, 2].

According to Definition 5, in a shortest path preserving labeling, shortest paths are exactly the preferred paths of the policy between the C and T nodes. This has the useful consequence that finding a graph where shortest path routing requires $\Omega(n)$ bits and showing a shortest path preserving labeling on it immediately yields proof on the incompressibility of the routing policy under consideration. Note that the definition permits that there can be other nodes $I \in V(G) \setminus (C \cup T)$ between which this property does not hold (e.g., the $z_{x,y}$ nodes in Fig. 3, for which the shortest path preserving property clearly does not hold).

Theorem 2: Valley-free routing with local preference is incompressible.

Proof: We prove this result, borrowing the idea in [10]. First, we show a family of graphs in which shortest path routing requires $\Omega(n)$ bits at some nodes. Then, we give a shortest path preserving labeling so that shortest paths from the constrained nodes to the target nodes precisely correspond to valley-free customer paths.[4]

We construct our graphs as follows. Start with $p \geq 2$ nodes, let these nodes make up the constrained node set C, and connect all these nodes into a full-mesh of peer links. To each $v_i \in C$, add $\delta \geq 2$ neighbors $z_{ij}, i \in \{1, \ldots, p\}, j \in \{1, \ldots, \delta\}$, and let the (z_{ij}, v_i) links be provider links. Finally, add δ^p target nodes, and connect these to the z_{ij} nodes according to the following rule: For each target node t, take the alphabet consisting of the symbols $(1, \ldots, \delta)$, construct a word of length p from this alphabet, and add a provider link from t to z_{ij} if the ith symbol in the word is exactly j. Fig. 3 gives an example.

Now, it is shown in [10] that min-hop routing in the above family of graphs requires $\Omega(n \log \delta)$ bits of storage space at the constrained nodes. Intuitively speaking, the idea is that there is an astronomical number of different graphs in this graph family, and to encode the shortest paths, the routing algorithm needs to differentiate among them, which requires many bits. What remains to be shown is that our link labeling is actually shortest path preserving.

Clearly, each node reaches any other node via a valley-free path. Furthermore, the shortest path from each constrained node $v_i \in C$ to any target node $t \in T$ is the unique two-hop customer path. At the same time, this path is the preferred one in our policy as well, as any other valley-free $v_i \to t$ path first crosses a peer link to some other $v_j \in C$ and so is less preferred by assumption. ■

[4]Strictly speaking, our proof is only valid if we limit the address size to no more than $\log n$ bits, but this restriction is easy to relax [11].

C. Valley-Free Routing With Local Preference and Shortest Path First

This model integrates the previous two models into a full-fledged policy routing architecture. In algebraic terms, this model arises as the lexicographic product of valley-free routing with local preference and min-hop routing: $\mathcal{A}_3 = \mathcal{A}_2 \times \mathcal{S}$. Since $\mathcal{A}_3$ contains $\mathcal{A}_2$ as a special case, the following result is immediate.

Theorem 3: Valley-free routing with local preference and shortest path routing is incompressible.

A straightforward proof would show that the incompressibility proof for $\mathcal{A}_2$ is also an incompressibility proof for $\mathcal{A}_3$.

D. Discussion

We have seen that, in terms of scalability, there is a huge gap between valley-free routing with and without local preferences. When actors are allowed to exercise their autonomy, imposing local preference rules on path selection, policy routing becomes incompressible. This is even without the incompressibility arising from shortest path routing, and even under the most simplistic and ubiquitous model for local preferences we could come up with.

An important final note is in order here. Certain model assumptions, like address size, are important for the validity of our proofs. If, for instance, we require that node degrees be bounded by a constant, then our proof stops working. These assumptions, however, do not really matter in the long run: The universal approach in [11], stating incompressibility of min-hop routing even for bounded degree and $c \log n$-bit address size, can be extended to our case easily.

More interesting is the case of global reachability and provider loops in valley-free routing. We have seen that if these principles hold, valley-free routing is compressible. However, when global reachability does not hold, then, as one easily sees, the labeling in Fig. 4(a) is shortest path preserving for the family of graphs given in the proof of Theorem 2. Here, there are some nodes (e.g., z_{12} and c_2) between which no valid path exists. This establishes the incompressibility of valley-free routing when the principle of global reachability is relaxed. Similarly, Fig. 4(b) shows a shortest path preserving labeling (with some extra nodes added) if provider loops are allowed, again implying incompressibility. It seems that positive compressibility results are pretty sensitive to the model perturbations, while negative incompressibility results are really robust.

VI. Relaxing Policy Requirements

Compact routing research is revolved around the desire to shrink the memory footprint of shortest path routing as much as possible. The general incompressibility of shortest path routing (Proposition 1) motivated the idea to reduce memory requirements at the price of introducing a moderate stretch on the paths. In this context, stretch can be seen as a relaxation of the shortest path criterion, by letting paths be slightly longer than the shortest ones, but keeping a strict upper bound on length increase by a stretch factor.

In the sequel, we ask to what extent the same trick can be played out for policy routing. In particular, we define several abstract notions of stretch for each of the incompressible policy

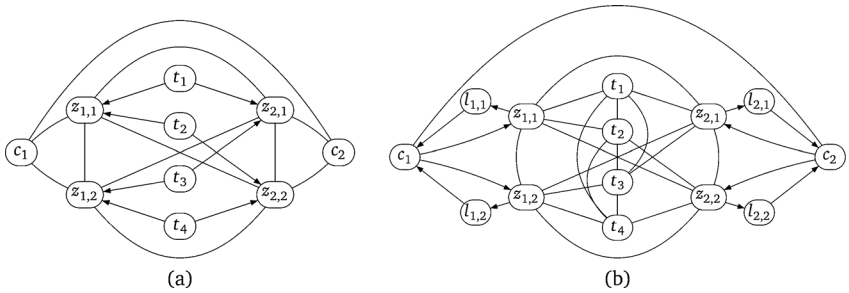

Fig. 4. Shortest path preserving labeling when (a) global reachability does not apply and (b) provider loops are allowed.

TABLE II
WEIGHT COMPOSITION IN BACKUP ROUTING (x STANDS FOR THE NUMBER OF STEPS)

$\oplus$	(x,c)	(x,r)	(x,p)
$(0,c)$	(x,c)	$(x+1,c)$	ϕ
$(0,r)$	(x,r)	$(x+1,r)$	$(x+1,r)$
$(0,p)$	(x,p)	(x,p)	(x,p)

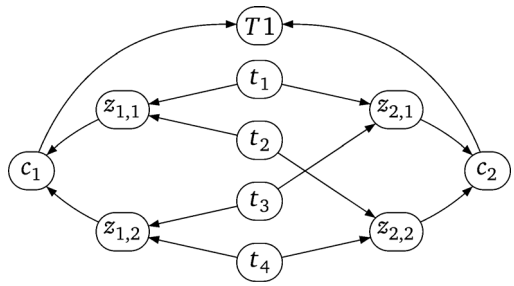

Fig. 5. Incompressibility proof for valley-free routing with local preference and step counting.

routing models identified in Section V, in the hope that this will yield a substantial reduction in storage requirements. First, we discuss valley-free routing with local preference, and then we also add the requirement of minimizing the path length.

A. Valley-Free With Local Preference and Step Counting

In Section V-B, we showed that combining valley-free routing with the simplest local preference rule renders the policy incompressible. Now, we ask whether relaxing either of the two components yields a compressible routing policy.

First, we relax the valley-free property. In particular, we allow the selected paths to contain a certain number of "steps," defined as $c-r$, $r-p$, or $r-r$ subpaths. This is a clear violation of the valley-free property, but our aim will be to put a strong constant upper bound on the number of such policy violations along any path. Steps were originally introduced in Backup Routing [28] to increase path diversity. We ask to what extent memory requirements can be decreased if we allow paths to contain at most a fixed number of steps while still maintaining local preference. The algebra is roughly modeled after [22].

Definition 6: Valley-free routing with local preference and step counting is defined as the algebra $\mathcal{A}_4 = (W, \phi, \oplus, \preceq)$, where we have the following.

- $W = \mathbb{N} \times \{c, p, r\}$, where the first component denotes the number of steps.
- $\oplus$ is as in Table II.
- $(v_1, w_1) \preceq (v_2, w_2)$ if $v_1 < v_2$, or $v_1 = v_2$ but $w_1 \preceq_{\mathcal{A}_2} w_2$ where $\preceq_{\mathcal{A}_2}$ is the precedence relation in $\mathcal{A}_2$.

Next, we give an exact definition for what we call stretch in this model.

Definition 7: A routing scheme is of stretch-k in $\mathcal{A}_4$ if for the weight $w(p) = (x, y)$ of path p it selects, $x \leq x^* + k$ and $y \preceq y^*$, where $w(p^*) = (x^*, y^*)$ is the weight of the preferred path.

Note that by global reachability there is a valley-free path between any two nodes, from which $x^* = 0$. For instance, path $p_2 - t_2$ in Fig. 1 is of stretch zero, but $p_2 - p_3 - t_2$ and $t_1 - p_2 - p_3$ are of stretch 1.

The following claim establishes that memory requirements cannot be reduced in valley-free routing with local preference, even if we allow arbitrary number of steps as policy violations.

Theorem 4: There is no stretch-k routing scheme over $\mathcal{A}_4$ for any constant k, which can be implemented with sublinear memory requirement at all nodes.

Proof: Consider the following modification of the graph construction in the proof of Theorem 2: Let the constrained nodes be connected to a common node $T1$ via a provider edge instead of a full peer-mesh (see Fig. 5). It is easy to see that the modified labeling is also shortest path preserving with respect to the c_i and t_i nodes since the c_i nodes cannot use the path through T_1 to reach the t_i nodes due to the local preference policy. As the graph does not contain peer links anymore, it cannot contain any steps, therefore any stretch-k routing scheme must encode exactly the preferred paths in $\mathcal{A}_2$. This establishes incompressibility by Theorem 2. ■

B. Valley-Free With Counting Local Preference Violations

So far, we have seen that there is no point in relaxing the valley-free property, as this does not yield reduction in memory requirements. Now, we relax the other component in $\mathcal{A}_2$, namely, local preference. In particular, we now strictly enforce the valley-free property, but we permit at most k local preference violations along the path. Unfortunately, this policy does not have a formal algebraic description because the composition operator cannot be defined properly. Therefore, we only give an informal description.

Definition 8: Valley-free routing with counting local preference violations is defined by the tuple $\mathcal{B} = (W, \phi, \preceq)$, where we have the following.

- $W = \{c, p, r\} \times \mathbb{N}$, where the second component denotes the number of local preference violations.

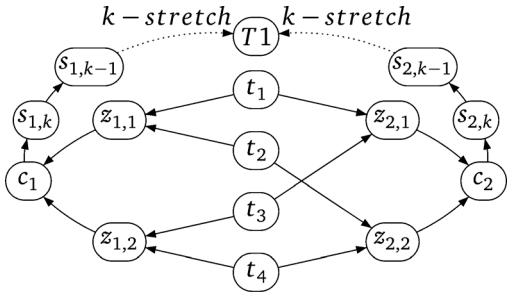

Fig. 6. Incompressibility proof for valley free routing with counting of local preference violations.

- The weight of a path (w, v) is defined as follows: w is composed of the first component of the link weights as of $\oplus_{\mathcal{A}_2}$ in Table I, and v is the number of times the path leaves a node on a p or r link even though the node has a valley-free path of weight c to the destination.
- $(w_1, v_1) \preceq (w_2, v_2)$ if $w_1 \preceq_{\mathcal{A}_2} w_2$, or $w_1 =_{\mathcal{A}_2} w_2$ but $v_1 \leq v_2$.

Definition 9: A routing scheme is of stretch-k in $\mathcal{B}$ if for the weight $w(p) = (x, y)$ of the paths p it selects, $x \preceq x^*$ and $y \leq y^* + k$, where $w(p^*) = (x^*, y^*)$ is the weight of the preferred path.

For instance, path $c_1 - p_1 - t_1 - p_2 - c_2$ is of stretch 1 in Fig. 1, as local preference is violated at p_1.

Theorem 5: There is no stretch-k routing scheme over $\mathcal{B}$ for any constant k, which can be implemented with sublinear memory requirement at all nodes.

Proof: Modify the graph construction in Theorem 4 by squeezing k nodes $s_{i,j}, j \in [1, k]$ between $T1$ and the constrained node c_i. In addition, connect c_i to $s_{i,k}$, $s_{i,j}$ to $s_{i,(j-1)}$: $j \in [2, k]$, and $s_{i,1}$ to $T1$ via a provider edge (see Fig. 6). Between any c_i and t_j, there are two valley-free routes—one is through a z-node, and the other is through $T1$. Again, the labeling is shortest path preserving with regard to the c_i and t_i nodes, and any suboptimal path goes through T_1 and, as such, of stretch larger than k for any k constant. ■

C. Valley-Free Routing With Shortest Path

We now take the next step and investigate to what extent the full-fledged model $\mathcal{A}_3$ admits a compact implementation when relaxing certain policy criteria. Clearly, relaxing the shortest path requirement would not change scalability since valley-free with local preference is incompressible in itself. Instead, we prove a stronger result here: Memory requirements cannot be reduced even if we completely let local preferences loose.

Definition 10: Valley-free routing with shortest path is defined by the algebra $\mathcal{A}_5 = \mathcal{A}_1 \times \mathcal{S}$.

Definition 11: A routing scheme is of stretch-k in $\mathcal{A}_5$ if for the weight $w(p) = (x, y)$ of the paths p it selects, $x \preceq x^*$ and $y \leq ky^*$, where $w(p^*) = (x^*, y^*)$ is the weight of the preferred path.

One easily checks, for instance, that the path $c_2 - p_2 - t_2 - p_3 - c_3$ in the sample network of Fig. 1 is of stretch 2 in $\mathcal{A}_5$.

Theorem 6: There is no stretch-k routing scheme over $\mathcal{A}_5$ for any constant k, which can be implemented with sublinear memory requirement at all nodes.

Proof: We observe that the graph construction of Theorem 5 gives an incompressibility proof for $\mathcal{A}_5$ as well since the path between the constraint and the target nodes through $T1$ has stretch greater than k. ■

VII. Conclusion

For the last couple of decades, compact routing research has focused on identifying the fundamental scaling limits of shortest path routing and constructing algorithms that meet these limits. Motivated by the realization that in most large networks where scalability indeed matters, it is not shortest paths along which traffic really flows, in this paper we asked to what extent compact routing theory can be generalized to policy routing.

In the first part of the paper, we defined three policy routing models of increasing complexity and gave a thorough theoretical scalability analysis. Then, we also asked to what extent the traditional memory–stretch tradeoff can be played out for policy routing. Our contributions in this regard are fourfold.

First, we gave a first report on routing policies that *inherently scale well*, like valley-free routing, and we also presented an incompressibility point marking the frontier between policies that can be implemented with compact routing tables and policies that do not. It is important to point out that our scalability analysis is valid in a broader sense. Our analysis is stated in terms of the local memory requirements of nodes needed to implement a policy, while a growing trend in compact routing is to study the *aggregate* memory requirement summed up along all nodes in the network. Here, the requirement for compressibility is $o(n^2)$ aggregate memory. Our results are valid under this model as well. In particular, valley-free routing remains compressible, while the other two models are incompressible.

Second, our models were deliberately chosen so that they are generic enough to warrant wide applicability while, at the same time, remain realistic enough to represent real-world policy routing architectures. Indeed, the three models introduced in Section III capture the first three consecutive stages in the decision process of the Border Gateway Protocol (BGP) rather closely. Consequently, our analysis reveals that *full-fledged BGP policy routing is incompressible*. Interestingly, it is not the valley-free property that renders BGP policy routing unscalable. In fact, global reachability, the absence of provider loops, plus the rather strict valley-free requirement impose a special hierarchy on the network, made up of the full-mesh of nodes (roughly corresponding to the Tier-1 service providers in the Internet) to which the rest of the nodes are connected through distinct customer cluster trees. The valley-free frame, representing this hierarchy, then lends itself readily to a compact addressing and routing scheme. Unscalability ensues when this delicate hierarchy is destroyed, either by a relaxation of the above assumptions or, more realistically, when we let nodes differentiate between the available valley-free paths through applying their own local preference settings. In this respect, we could say that incompressibility of BGP policy routing is the "price of anarchy" in the Internet.

Third, our analysis showed that the incompressibility proofs for policy routing are rather strict, in the sense that the memory–stretch tradeoff, which proved quite fruitful for shortest path routing, cannot be invoked to alleviate local

storage requirements. We examined three different abstract notions of stretch, one where we allowed the valley-free criterion to be violated, one where we allowed for a certain number of local preference violations, and one where we used the traditional notion of shortest path stretch. It turns out that none of these stretch definitions yield compact routing tables. In this context, future work involves extending this analysis to other abstract notions of stretch.

Fourth, our scalability analysis might give clues to designing future routing policies that do scale well [29]. Not just that such a policy must admit scalable implementation, but it must be incentive-compatible at the same time. This seems an intriguing future research question.

REFERENCES

[1] A. Singla, P. B. Godfrey, K. Fall, G. Iannaccone, and S. Ratnasamy, "Scalable routing on flat names," in *Proc. CoNEXT*, 2010, Art. no. 20.

[2] R. Agarwal, P. B. Godfrey, and S. Har-Peled, "Approximate distance queries and compact routing in sparse graphs," in *Proc. IEEE INFOCOM*, 2011, pp. 1754–1762.

[3] Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith, "S4: Small state and small stretch routing protocol for large wireless sensor networks," in *Proc. 4th NSDI*, Apr. 2007, p. 8.

[4] V. Jacobson *et al.*, "Networking named content," in *CoNEXT*, 2009, pp. 1–12.

[5] T. Koponen *et al.*, "A data-oriented (and beyond) network architecture," in *Proc. SIGCOMM*, Aug. 2007, pp. 181–192.

[6] A. Fabrikant, A. Luthra, E. N. Maneva, C. H. Papadimitriou, and S. Shenker, "On a network creation game," in *Proc. ACM PODC*, 2003, pp. 347–351.

[7] H. Levin, M. Schapira, and A. Zohar, "Interdomain routing and games," in *Proc. ACM STOC*, 2008, pp. 57–66.

[8] A. Fabrikant and C. H. Papadimitriou, "The complexity of game dynamics: BGP oscillations, sink equilibria, and beyond," in *Proc. ACM-SIAM SODA*, 2008, pp. 844–853.

[9] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 232–243, Apr. 2002.

[10] P. Fraigniaud and C. Gavoille, "Memory requirement for universal routing schemes," in *Proc. ACM PODC*, 1995, pp. 223–230.

[11] C. Gavoille and S. Perennes, "Memory requirement for routing in distributed networks," in *Proc. ACM PODC*, 1996, pp. 125–133.

[12] M. Thorup and U. Zwick, "Compact routing schemes," in *Proc. ACM SPAA*, 2001, pp. 1–10.

[13] L. Cowen, "Compact routing with minimum stretch," in *Proc. ACM-SIAM SODA*, 1999, pp. 255–260.

[14] G. Huston, "Interconnection, peering, and settlements," in *Proc. INET*, 1999.

[15] M. Caesar and J. Rexford, "BGP routing policies in ISP networks," EECS Department, University of California, Berkeley, CA, USA, Tech. Rep. UCB/CSD-05-1377, 2005 [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2005/6507.html

[16] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 681–692, Dec. 2001.

[17] S. Dibenedetto, C. Papadopoulos, and D. Massey, "Routing policies in named data networking," in *Proc. ACM SIGCOMM Workshop Inf. Centric Netw.*, 2011, pp. 38–43.

[18] C. Gavoille, "Routing in distributed networks: Overview and open problems," *ACM SIGACT News*, vol. 32, no. 1, p. 52, 2001.

[19] D. Papadimitriou, "Compact routing: Challenges, perspectives, and beyond," TRILOGY Future Internet Summer School, Louvain-la-Neuve, Belgium, Tech. Rep., 2009 [Online]. Available: http://typo3.trilogy-project.eu/fileadmin/publications/Other/Papadimitriou-CompactRouting.pdf

[20] G. Rétvári, A. Gulyás, Z. Heszberger, M. Csernai, and J. J. Bíró, "Compact policy routing," *Distrib. Comput.*, vol. 26, no. 5–6, pp. 309–320, Sep. 2012.

[21] X. Yang, D. Clark, and A. Berger, "NIRA: A new inter-domain routing architecture," *IEEE/ACM Trans. Netw.*, vol. 15, no. 4, pp. 775–788, Aug. 2007.

[22] J. Sobrinho, "Network routing with path vector protocols: Theory and applications," in *Proc. SIGCOMM*, 2003, pp. 49–60.

[23] CAIDA, La Jolla, CA, USA, "CAIDA project," [Online]. Available: http://www.caida.org/

[24] J. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 541–550, Aug. 2002.

[25] T. Griffin and J. Sobrinho, "Metarouting," in *Proc. SIGCOMM*, 2005, pp. 1–12.

[26] A. Gurney and T. Griffin, "Lexicographic products in metarouting," in *Proc. IEEE ICNP*, 2007, pp. 113–122.

[27] P. Fraigniaud and C. Gavoille, "Routing in trees," in *Proc. ICALP*, 2001, pp. 757–772.

[28] L. Gao, T. G. Griffin, and J. Rexford, "Inherently safe backup routing with BGP," in *Proc. IEEE INFOCOM*, 2001, vol. 1, pp. 547–556.

[29] A. Seehra *et al.*, "A policy framework for the future internet," in *Proc. HotNets-VIII*, 2009.

András Gulyás received the M.Sc. and Ph.D. degrees in informatics from the Budapest University of Technology and Economics (BME), Budapest, Hungary, in 2002 and 2008, respectively.

Currently, he is a Research Fellow with the Department of Telecommunications and Media Informatics, BME. His research interests are complex and self-organizing networks, network calculus and software defined networking.

Gábor Rétvári (S'03–M'05) received the M.Sc. and Ph.D. degrees in electrical engineering from the Budapest University of Technology and Economics (BME), Budapest, Hungary, in 1999 and 2007, respectively.

He is now a Senior Research Fellow with the High Speed Networks Laboratory, Department of Telecommunications and Media Informatics, BME. His research interests include QoS routing, traffic engineering, and the networking applications of computational geometry and the mathematical theory of network flows. He is a Perl expert, maintaining numerous open source scientific tools written in Perl, C, and Haskell.

Zalán Heszberger (S'99-A'01–M'14) received the M.Sc. and Ph.D. degrees in electrical engineering from the Budapest University of Technology and Economics (BME), Budapest, Hungary, in 1997 and 2007, respectively.

Currently, he is an Associate Professor with the Department of Telecommunications and Media Informatics, BME. His main research interests are future Internet technologies and complex networking. Currently, he is working on clean-slate design Internet routing and network management algorithms.

Rachit Agarwal received the Ph.D. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2013.

He is now a Postdoctoral Fellow with the University of California, Berkeley, CA, USA. His research spans networked systems and theory.