On the Memory Requirement of Hop-by-hop Routing: Tight Bounds and Optimal Address Spaces

Attila Kőrösi, András Gulyás*, Zalán Heszberger*, József Bíró, Gábor Rétvári
MTA-BME Information Systems Research Group, Budapest University of Technology and Economics,
H-1117 Budapest, Magyar tudósok krt. 2, Hungary
*Supported by the János Bolyai Fellowship of the Hungarian Academy of Sciences
Email: {korosi, gulyas, heszberger, biro, retvari}@tmit.bme.hu

***Abstract*—Routing in large-scale computer networks today is built on hop-by-hop routing: packet headers specify the destination address and routers use internal forwarding tables to map addresses to next-hop ports. In this paper we take a new look at the scalability of this paradigm. We define a new model that reduces forwarding tables to sequential strings, which then lend themselves readily to an information-theoretical analysis. Contrary to previous work, our analysis is not of worst-case nature, but gives verifiable and realizable memory requirement characterizations even when subjected to concrete topologies and routing policies. We formulate the optimal address space design problem as the task to set node addresses in order to minimize certain network-wide entropy-related measures. We derive tight space bounds for many well-known graph families and we propose a simple heuristic to find optimal address spaces for general graphs. Our evaluations suggest that in structured graphs, including most practically important network topologies, significant memory savings can be attained by forwarding table compression over our optimized address spaces. According to our knowledge, our work is the first to bridge the gap between computer network scalability and information-theory.**

I. Introduction

Throughout its several decades of history, the Internet has evolved from an experimental academic network to a global communications infrastructure. Most of the architectural transitions that have taken place in the background, from the ARPANET protocols to IP, from classful addressing to classless, from IP version 4 to version 6, were (and are) largely fueled by concerns regarding the ability of the network, and the underlying design principles, to accommodate future growth.

Today, most computer networks are built on the distributed hop-by-hop routing paradigm. Routers maintain forwarding tables to associate incoming packets with next-hop routers based on the destination address encoded in the packet headers, and subsequent routers use the same mechanism to deliver packets hop-by-hop to the intended target. Correspondingly, routers must keep enough information in internal memory to be able to forward any packet, with any destination address, to the right next-hop. Unsurprisingly, it is precisely this point where Internet scalability issues are manifesting themselves most visibly [1], [2], [3], [4]: as the routed IP address space grows so do the forwarding tables and when routers run out of memory (or TCAM space) major outages spark throughout the Internet, as it happened in August 2014, the infamous 512kday [5].

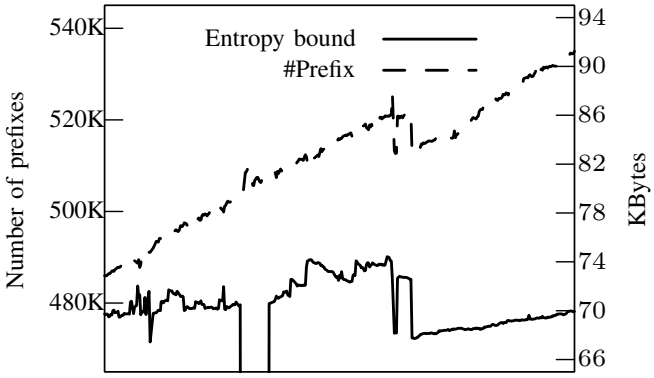


Fig. 1: Number of IPv4 prefixes and information-theoretical entropy bound on `rtr2.vh.hbone.hu` during 18 months in 2014 and 2015.

We created an Internet data plane measurement infrastructure to understand the long-term trends affecting Internet scalability[1]. We have collected roughly two dozen IPv4 forwarding tables every day during 2014 and 2015 from operational IP routers located in the Default Free Zone. Throughout this time, the number of entries in the forwarding tables has grown more than 11 percent to well over half a million (see Fig. 1). Strikingly, our statistics at the same time also indicate that the effective information content stored in the forwarding tables, in terms of the information-theoretic entropy bound defined in [6], has remained relatively stable (increased by only 0.5%), suggesting that the memory footprint of the *compressed representation* of these forwarding tables was constantly around 70 Kbytes within this time frame[2]. If the entropy bound is consistently and robustly invariant to the network size, as our measurements seem to indicate, then this can potentially mask the expansion of the Internet from operators and alleviate rising scalability concerns for the time coming [2].

The systematic study of the memory implications of large-scale routing was pioneered by Kleinrock and Kamoun in their seminal paper [8]. They pointed out that scalability is

[1]We publish browsable daily statistics and downloadable data sets at the *Internet Routing Entropy Monitor* website, see http://lendulet.tmit.bme.hu/fib_comp.

[2]See [6], [7] on how to compress forwarding tables to entropy bounds without sacrificing the performance of lookups and updates.

the central design requirement for very large networks and concluded that some form of forwarding state compression is inevitable. On the traces of McQuillan [9] they proposed a hierarchical clustering scheme, whereas nodes keep detailed forwarding information only about nearby nodes and apply gradually more coarse-grained state aggregation as distance increases. This way, they realized significant forwarding table reduction at the price of only limited increase in path length[3]. Underlying Kleinrock's result is the observation that node addresses can encode substantial knowledge about the network topology and this knowledge can be readily leveraged to shrink forwarding tables. Hierarchical routing still lives on in today's Internet routing architecture and it remains the only viable option, with well-known scaling properties, to engineer large-scale address spaces to our days (see e.g., RNR [12], GSE [13], Nimrod [14], ISLAY [15]).

Hierarchical routing has some intrinsic limitations. First, node addresses serve as "names" that encode the entire upstream cluster hierarchy, which causes problems with handling topology changes, failure recovery, and renumbering [16]. A *flat address space* for hierarchical routing, on the other hand, would introduce an extra name-to-address mapping step in all communication sessions [12]. Second, hierarchical routes are decidedly sub-optimal (i.e., worse than the best available path in the topology), while in most real-life scenarios *optimal routing* is a must. In Internet inter-domain routing, mistaking a customer route for a much more costly provider route or a trusted third-party for an unreliable intermediary can be detrimental, so much so that a suitable notion of path length dilatation cannot even be defined in this setting [17]. Unfortunately, hierarchical addressing for optimal routing is inefficient unless the underlying topology happens to be a tree.

The main goal of this paper is to *(re)launch the systematic study of the memory requirement of hop-by-hop routing over flat addresses and optimal routing*. We take a principled approach: we introduce certain entropy-like measures to describe the compressed size of forwarding state and we provide an information-theoretic analysis to uncover the fundamental scaling properties of large-scale hop-by-hop routing.

First, we give a model that *translates problems related to routing scalability to the language of information-theory.* Forwarding tables are modeled as sequential strings, admitting tight memory requirement characterizations using Shannon's entropy measures and standard data compression techniques. As a main contribution, *we formulate the problem of optimal address space design as a combinatorial optimization problem*, with the objective to minimize the general memory requirement of routing tables in terms of the above entropy measure.

Second, *we present asymptotically optimal solutions to the address space design problem over well-known graph families* when forwarding paths are selected using shortest-path routing. Using pure information-theoretic arguments, we are able to reproduce most of the space bounds obtained in the compact routing literature using piecemeal addressing schemes and analysis. For general graphs, where solving the problem to optimality promises intractable, we present *a fast heuristic address space design algorithm that runs in polynomial time*.

Finally, *we empirically evaluate the routing table compression ratio achievable by our techniques*. Given that our entropy measure gives general, tight, and firm *lower-bounds* on the memory footprint attainable by *any* forwarding table encoding scheme (e.g., [18]), we are able to show that essentially *no forwarding table compression can be achieved on large-scale structureless graphs* (e.g., on Erdős-Rényi random graphs). This finding is analogous to the well-known lower bound by which no scheme can guarantee sublinear forwarding state in general graphs [19] On the positive side, we find that most practically relevant network topologies are *structured* and *admit significant routing table compression, even without a specifically designed address space*. If, furthermore, *we are allowed to pick node addresses* then, using our heuristic address space designs, *the memory footprint of hop-by-hop routing can be reduced even further*: for highly structured graphs the space reduction is in the orders-of-magnitude range, while on large-scale Internet models [20] routing tables can be compressed below 25% of the uncompressed size. We stress that the routes in these evaluations are strictly shortest, whereas all previous approaches producing similar space reduction diverged from shortest paths and introduced significant path-dilation (stretch) [20], [21], [22]. This result suggests strong potential for practical forwarding table encoding schemes that can compress to entropy (e.g., [6], [7], [23], [24]).

The rest of this paper is structured as follows. Section II gives a hands-on introduction to data compression theory, based on which Section III introduces three information-theoretic models for routing scalability and formulates the address space design problem. Section IV provides tight space bounds for many important families of graphs (trees, grids, torii, hypercubes) over min-hop routing and discusses optimal address space design for general graphs. Section VI contains applicability issues with respect to real world network models and the Internet and Section VII concludes our work.

II. A Primer on Data Compression

Information theory is concerned with the storage, encoding, and transmission of text messages and the quantification of the information content thereof. A key measure in information theory is *empirical entropy*, the average number of bits needed to encode one symbol of a given text, which directly transforms into bounds on the efficiency of *any* data compression scheme [25], [26].

There are various approaches to represent the empirical entropy for an input text, relative to a model for the source that generates it. Throughout the paper, we assume that this model is *static*, that is, the compression scheme can use only information that is available from the source *a priori* and it does not depend on the particular instance of data that is being encoded. Of course, finding the best model for some problem domain is the real difficult part in information theory.

[3]Uncovering the path-length–memory trade-off later developed into a separate research field, compact routing [10], [11].

A. No-information Model

Suppose that we do not possess any explicit knowledge on the input string s apart from its length n and the alphabet $\Sigma = \{c_1, c_2, \ldots, c_\delta\}$, $|\Sigma| = \delta$ it is defined on. Then, encoding s is equivalent to being able to distinguish between any two of the possible δ^n strings we can get as input, which needs at least $I(s) = \log(\delta^n) = n \log \delta$ bits[4]. $I(s)$ is called the *information-theoretical lower bound*, referring to that we cannot hope for any compression beyond $I(s)$ unless the source discloses some further knowledge on the strings it generates.

B. Zero-order Model

Suppose now that, beyond the alphabet Σ and length n, we also know the number of occurrences n_c of each symbol $c \in \Sigma$ in s, but we do not have any *a priori* knowledge on the way symbols follow each other. In most cases we can use this additional knowledge to compress s beyond the information-theoretical lower bound. The best compression rate is bounded by the *zero-order empirical entropy* of s, defined as

$$H_0(s) = \sum_{c \in \Sigma} \frac{n_c}{n} \log \frac{n}{n_c} \ . \tag{1}$$

Trivially, we have $H_0(s) \leq \log \delta$ with $H_0(s) = \log \delta$ if and only if the empirical symbol distribution in s is uniform (i.e., $\frac{n_{c_1}}{n} = \frac{n_{c_2}}{n} = \ldots$).

On the one hand, the *zero-order entropy bound* $nH_0(s)$ is an upper limit on the number of bits needed to encode s as there are well-known compression schemes (e.g., Huffman coding, arithmetic coding) that attain roughly this size (the real bound is $nH_0(s) + o(n)$ bits). On the other hand, $nH_0(s)$ is also a firm lower bound, in that no encoding scheme can attain smaller compressed size under a static zero-order model.

As we do not have knowledge about the way symbols are laid out in s, apart from the relative frequencies $\frac{n_c}{n}$, we are confined to conservatively believe that the symbols follow each other randomly. Consequently, *the zero-order storage bound does not depend on the actual order in which symbols appear* in s (c.f., (1)). Thus, using a zero-order compressor the string `mississippi` and the anagram `miiissssspp` encode to the same size, roughly 20 bits ($H_0 \sim 1.823$ in both cases).

C. Higher-order Models

In practice, subsequent symbols in a text often do not follow each other haphazardly. Rather, certain symbols appear more frequently in certain contexts and almost never in others, like in normal English text the letter `q` is almost certain to be succeeded by the letter `u` while basically never by the letter `c` or `h` [27]. In general, for any $k > 0$ integer define the *k-context* for the symbol $s[i]$ appearing at some position $i \in [k+1, n]$ in s as the k-long string $s[i-k] \ldots s[i-1]$ immediately preceding $s[i]$ and for any k-context $q \in \Sigma^k$ let n_{qc} denote the number of times q is followed by symbol c in s. Then, the *k-order empirical entropy*

$$H_k(s) = \sum_{c \in \Sigma} \frac{n_c}{n} \sum_{q \in \Sigma^k} \frac{n_{qc}}{n_c} \log \frac{n_c}{n_{qc}} \tag{2}$$

gives a lower bound to the output size of any text compressor that encodes each symbol with a codeword that depends only on the k-context preceding the symbol and the symbol itself [28], [29]. $H_k \leq H_{k-1}$ and it is generally held that H_k converges to the "real" empirical entropy for large k. In practice usually taking $k = 4$ or 5 is enough and H_k decreases very slowly even after $k = 1$. Therefore, we shall use $k = 1$ henceforth.

As opposed to the zero-order model, *in a higher-order model the arrangement of the symbols in s* does *count*: the more organized the string the better the prediction of a symbol from its k-context and so the smaller the k-order entropy and the size of the compressed string. For instance, for the string `mississippi` $H_1 \sim 0.8$ but for the visibly more regular anagram `miiiissssspp` we get only $H_1 = 0.6$. This suggests that in a setting where we are to a certain extent free to choose the arrangement of the symbols in some string (as will be the case below) we should strive for more order, which would then directly translate into better compressibility through the notion of higher-order entropy. At the extreme, if we can reorder a string into a few runs of identical symbols (like in the example `miiissssspp`) we realize maximum compression by simple run-length encoding.

D. Compressed Data Structures

In the context of this paper we do not only want to squeeze data into small memory but we also want to execute certain operations in place, most importantly fast *random access* to any position in the string and, possibly, arbitrary *updates* to the content as well [6]. Traditional data compression schemes, like Huffman coding or run-length encoding, do not support such operations without first decompressing the data. Thanks to recent advances on compressed data structures, however, there are now a well-tested suite of compression schemes that allow fast operations right on the compressed form.

Static compressed self-indexes implement fast random access to the compressed string but no updates (without a complete rebuild from scratch). When the size of the alphabet is small, say, $\delta = O(\log n)$, generalized wavelet trees [30] attain $nH_0(s) + o(n)$ bits of space and $O(1)$ random access, while the schemes in [28] and [31] attain $nH_k(s) + o(n \log \log n)$ bits space for any $k = o(\log n)$ with random access in $o(\log \log \log n)$ time. *Dynamic* compressed indexes permit updates to any position as well: the scheme in [32] attains $nH_0(s) + o(n \log \log n)$ bits space and random access and update in $O(\log n \log \log n)$ time. For precise definitions and generic storage bounds, see the surveys [33], [34], [35], [36].

III. Hop-by-hop Routing Model

We adopt the model of static oblivious routing functions for compact routing from [19].

[4] All logarithms in this paper are taken to the base 2 and we assume $0 \log 0 = 0$. For brevity, we do not differentiate between $\lceil \log n \rceil$ and $\log n$ unless otherwise noted.

Let $G(V, E)$ be a simple connected undirected graph on n nodes. We presume a flat address space on V: each node $v \in V$ is labeled with a globally unique integer $id(v) \in [1, n]$. Note that ids are the only "addresses" we use to identify nodes and as such they are of global scope. (The question whether we are allowed to change node ids will prove essential, we shall return to this crucial question soon.) In addition, each outgoing port $(v, u) \in E$ of each node $v \in V$ is also labeled with a locally unique port id $port(v, u) \in [1, \delta_v]$, where δ_v denotes the node degree of v. Port ids are local and hence can be selected arbitrarily[5] in the range $[1, \delta_v]$.

We further presume that some routing policy (e.g., shortest-path, min-hop path, valley-free) for each node has been fixed in advance and packet forwarding must strictly obey the paths emerging from this policy. As far as our model is concerned, however, we do not assume any particularity about the routing policies themselves whatsoever, apart from that *(i)* the policy is such that it lends itself to a distributed implementation (see [38] for examples where this requirement does not hold) and *(ii)* at each node it assigns a single, well-defined output port to each other node that packets destined to that node must be forwarded via.

Packets contain a header with the id of the destination node. This is then fed to the local routing function s_v of each node v along the forwarding path that returns the output port to pass the packet on. The routing function $s_v : [1, n] \mapsto [1, \delta_v]$ maps a destination node id to the corresponding outgoing port. We suppose that each node $v \in V$ is aware of its own id (this immediately imposes $\log(n)$ bits lower space bound for storing s_v that we shall omit from here onwards) and hence can identify packets destined to itself, so we shall usually set $s_v(id(v))$ to an arbitrary port id.

It is convenient to think of s_v as a string of length n on the alphabet $\Sigma_v = [1, \delta_v]$, so that the symbol $s_v[i]$ at position $i \in [1, n]$ gives the output port to be used to forward packets towards the node whose id is i. Consequently, a forwarding decision in this setting reduces to a random access on s_v (which most compressed string self-indexes support out-of-the-box, see Section II-D) and modifications are simple string updates. See a sample network and routing function in Fig. 2.

IV. Memory Requirement for Hop-by-hop Routing

The main concern in this paper is to find lower and upper bounds on the number of bits needed to encode the routing function at each node. Since routing functions boil down to simplistic strings in our model, we can exert the entire toolset of information-theory to infer space bounds. As usual in information theory, however, the answer depends on the actual model we choose for designing the routing functions, as discussed next.

[5]The compact routing literature distinguishes between the designer port model where port ids are assigned by the designer, and the fixed-port model where port ids are set by an adversary [37]. In our model these two cases coincide.

A. Graph Independent Case

Suppose that the source does not reveal any further knowledge on the graph and the selected forwarding paths other than the number of nodes n. Then, all we know is that the routing functions will comprise n symbols, each coming from the alphabet $\Sigma_v = [1, n-1]$. We are bound to assume $|\Sigma_v| = n-1$, since the only upper limit on the alphabet size that holds in any simple graph is that the maximum degree is at most $n-1$.

It is then easy to see that the graph independent case essentially maps to the no-information model of Section II-A, and hence the information-theoretic lower bound of $I_v = I(s_v) \sim n \log n$ bits is the best space bound we can hope for in this setting. It turns out that this is also a very generic *lower bound*, since Gavoille and Pérennès showed that *any* routing scheme realizing shortest path routing on graphs of size n necessarily stores $\Omega(n \log n)$ bits of information at some $\Omega(n^\epsilon)$ nodes, for any $0 < \epsilon < 1$ [19]. This is bad news, as superlinearly scaling memory for mere packet forwarding, in a network growing as rapidly as the Internet, would put routers under endless memory stress [2].

B. Name Independent Case

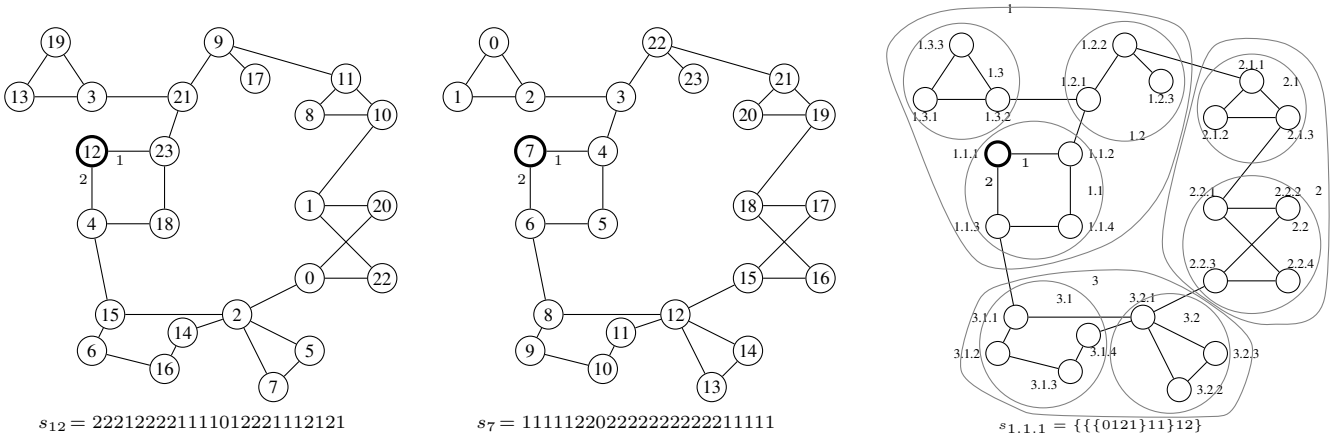
A way out of the above trap would be a model where the source reveals further information to the designer. Could we design more compact routing functions if we knew the input graph, for instance? As mentioned previously, in the compact routing literature it is common to distinguish between the cases when node ids are fixed in advance and it is beyond our reach to change them, and when we are free to assign node ids in the design phase. The former, called the name independent model [10], [37], [39], is discussed next.

In this model, we get the input graph G (along which comes n and the routing function alphabet $\Sigma_v = [1, \delta_v]$ for each $v \in V$), the required next-hop ports for each node pair, and the node ids $id(v) : v \in V$, but these ids do not communicate any further information apart from that they uniquely identify the nodes. We could as well view this as if node ids were chosen by an adversary and so the memory at each node should be large enough to store any possible routing function arising over any permutation of ids. See Fig. 2a.

The name independent case in this setting maps naturally to the zero-order model of information-theory (Section II-B): we can compute the zero-order next-hop port distribution $n_i/n : i \in [1, \delta_v]$ for each port i of each $v \in V$ but no higher-order statistics, as these depend on the assignment of node ids that is beyond our control. As shall be seen, in some cases (like complete graphs or stars) this knowledge is not enough to compress the routing functions below the information-theoretic lower bound, but in many practically relevant cases significant memory savings can be realized.

The following gives a tight bound on the attainable compression in the name independent case.

Theorem 1. *Given a graph G on n nodes with each node assigned a unique id in $[1, n]$ under the* name independent



(a) random address space (b) optimal address space (c) hierarchical address space

Fig. 2: Sample graph from Kleinrock's original paper [8] with port ids and routing function for the thick node over different address spaces; (a) random node ids (average zero-order routing function entropy $\bar{H}_0 = 1.1$ bit); (b) optimal node ids found by brute force search ($\bar{H}_1 = 0.355$ bit); and (c) hierarchical node addresses and forwarding table (omitting self-entries) as of [8]. Observe that nearby nodes are mapped to close-to-each-other node ids in the optimal address space (b), which translates into three-fold space reduction compared to a random address space of (a). Hierarchical routing (c) yields even smaller tables but does not reproduce the optimal paths.

model, *encoding the routing function* $s_v : v \in V$ *needs* $nH_0(v) + o(n)$ *bits memory, where*

$$H_0(v) = H_0(s_v) = \sum_{i \in [1,\delta_v]} \frac{n_i}{n} \log \frac{n}{n_i} \tag{3}$$

and n_i *denotes the number of times output port id* $i \in [1, \delta_v]$ *appears as a next-hop port in* s_v.

Proof. Easily, $nH_0(v) + o(n)$ bits is enough to store s_v, so we only need to show that this is a lower bound as well. We observe that, depending on how the adversary assigns node ids, the next-hop port id 1 can appear at exactly $\binom{n}{n_1}$ positions in s_v, port id 2 can appear at $\binom{n-n_1}{n_2}$ positions, etc., putting the number of distinct routing functions we need to identify to

$$\binom{n}{n_1}\binom{n-n_1}{n_2}\cdots\binom{n-\ldots-n_{\delta_v-1}}{n_\delta} = \frac{n!}{\prod_{i=1}^{\delta_v} n_i!} \ .$$

Using the Stirling formula $\ln n! \sim n \ln n - n + O(\ln n)$, we get that we need $\log \frac{n!}{\prod_i n_i!} = nH_0(v) + o(n)$ bits to identify each possible routing function. □

C. Name Dependent Case

In the final model we discuss in this paper, we again require the graph G to be fixed but we are now free to assign node ids. This case is usually referred to as the name dependent model in compact routing [10] and, as one easily checks, it maps to the higher-order models of information-theory (Section II-C).

Indeed, the sequence of next-hop ports in the routing functions is completely determined by the node id assignment, which in this case may not be some arbitrary permutation like in the zero-order model. Rather, the id space could be carefully optimized in order to encode maximum information about the underlying topology into the ids proper and so the (otherwise flat) node id space can function as a full-fledged *address space* for the network at hand. This in turn might make it possible to compress routing functions more efficiently, by the fact that a carefully chosen node id space would transform into smaller higher-order entropy.

The corresponding space bounds are then as follows:

Theorem 2. *Given a graph* G *on* n *nodes with each node assigned a unique id in* $[1, n]$ *under the* name dependent model, *encoding the routing function* $s_v : v \in V$ *needs* $nH_k(v) + o(n)$ *bits memory for any* $k > 0$ *integer, where*

$$H_k(v) = H_k(s_v) = \sum_{q \in [1,\delta_v]^k} \sum_{i \in [1,\delta_v]} \frac{n_{qi}}{n} \log \frac{n_i}{n_{qi}} \ , \tag{4}$$

n_i *is as previously and* n_{qi} *is the number of times port* i *succeeds the* k*-long port sequence* $q \in [1, \delta_v]^k$ *in* s_v.

The proof is similar to that of Theorem 1 and omitted here for brevity.

The question arises then how to assign node ids to minimize the above higher-order entropy bound, that is, how to design the address space? A possible strategy would be to map (topologically) adjacent nodes to consecutive node ids, based on the intuitive idea that from a far-away point in the graph such adjacent nodes would most probably be reached through similar next-hop ports, which would then allow to use the context of some node (i.e., the next-hop ports for the nodes mapped to preceding node ids) to guess the next-hop port for this node. Later, we show that this strategy yields significant memory savings in many networks.

This discussion is formalized in the following problem statement, the *optimal address space design problem*:

TABLE I: Entropy ($\bar{H}_0$, $\bar{H}_1$) and compression ratio (η_0, η_1) under the name independent and the name dependent model for various well-known families of graphs, along with the node enumeration strategy that produces the optimal address space and the corresponding bounds from compact routing. As the latter are usually given in terms of the whole routing function size, we computed the average per node (i.e., divided the results by n). In graphs where multiple shortest paths exist between many node pairs, for $\bar{H}_0$ we specify the lower and upper bound over an optimistic and a pessimistic choice for the next-hops and for $\bar{H}_1$ we specify only the optimistic setting.

	Name independent		Name dependent			
Graph class	$\bar{H}_0$	η_0	$\bar{H}_1$	η_1	Optimal address space	Compact routing
complete graph	$\log n$	1	0	0	arbitrary	$O\left(\frac{\log n}{n}\right)$ [19]
star	$\frac{\log n}{n}$	1	0	0	arbitrary	$O\left(\frac{\log n}{n}\right)$ [10]
d-grid	$\frac{\log e}{2} \longleftrightarrow \log(2d)$	$\frac{\log e}{2\log(2d)} \longleftrightarrow 1$	$d\frac{\log n}{n}$	$o(1)$	coordinate-wise	$d\frac{\log n}{n}$ [40]
path graph	$\frac{\log e}{2} \sim 0.72$	~ 0.72	$2\frac{\log n}{n}$	$o(1)$	coordinate-wise	$2\frac{\log n}{n}$ [40]
d-torus	$1 \longleftrightarrow \log(2d)$	$\frac{1}{\log(2d)} \longleftrightarrow 1$	$(d+1)\frac{\log n}{n}$	$o(1)$	coordinate-wise	$d\frac{\log n}{n}$ [40]
ring	1	1	$2\frac{\log n}{n}$	$o(1)$	coordinate-wise	$2\frac{\log n}{n}$ [40]
d-hypercube	$2 \longleftrightarrow \log d$	$\frac{2}{\log d} \longleftrightarrow 1$	$d\frac{\log n}{n}$	$o(1)$	lexicographical	$d\frac{\log n}{n}$ [40]
n-tree	$\frac{\log n}{n} \longleftrightarrow \frac{\log e}{2}$	NA	$2\frac{\log n}{n}$	NA	post-order	$3\frac{\log n}{n}$ [41], $O(1)$ [11]

Definition 1. *Given routing functions $s_v : v \in V$, $|V| = n$ and any $k = o(\log n)$, find the permutation π of node ids that minimizes the mean k-order entropy of the forwarding tables:*

$$\min_{\pi} \frac{1}{n} \sum_{v \in V} H_k(\pi(s_v)) \ . \tag{5}$$

Alternatively, one might aim for minimizing the maximal entropy instead: $\min_\pi \max_v H_k(\pi(s_v))$. Note that in both cases the permutation π must be globally optimal, that is, we must design a single node id allocation that reduces the entropy at *each* node simultaneously. An optimal address space design, as found by solving (5) by a brute-force search algorithm, is given in Fig. 2b.

V. Analysis and Algorithms

Next, we turn to analyze the memory requirement of hop-by-hop routing. First, we present the results for some well-known graph families and next we give a heuristic characterization for general graphs.

A. Asymptotically Optimal Results on Well-known Graphs

Computing the entropy for a routing function is already difficult for the zero-order setting, let alone in a higher-order model. Therefore, it seems hopeless to seek for exact general characterizations. Rather, below we concentrate on instances for which the problem *can* be solved: highly symmetric graphs from some well-known graph families. Most of the results below have already been known from the compact routing literature, we still reproduce these here to demonstrate the basic algorithmic techniques and to motivate our address space optimization heuristics (see the next section).

Below, we stick to the simplest possible case: shortest-path routing over unit cost graphs[6]. The analysis will be for the average case: for each graph class we specify the average (over all nodes) empirical zero-order entropy $\bar{H}_0 = {}^1\!/_n \sum_v H_0(v)$; the average first-order entropy $\bar{H}_1 = {}^1\!/_n \sum_v H_1(v)$ attainable over an optimal address space; and the compression ratios $\eta_0 = {}^{n\bar{H}_0}\!/_{\bar{l}}$ and $\eta_1 = {}^{n\bar{H}_1}\!/_{\bar{l}}$ where $\bar{l}$ denotes the average information-theoretic lower bound and $n\bar{H}_0$ ($n\bar{H}_1$) is the mean forwarding table size in bits. The results are in Table I.

Full-meshes (complete graphs) and hup-and-spoke networks (stars) are amongst the most renowned topologies in telecommunications. In the name independent case (recall, we cannot modify node ids in this setting) we get no compression at all for these topologies. For complete graphs in particular, we cannot even get below the theoretical lower bound of [19], [42]. In the name dependent case however, when we are free to assign node ids, entropy bounds fall to zero and we get infinite compression. Note, however, that $\bar{H}_1 = 0$ is purely theoretic, as for a node at least storing its own id requires $\log n$ bits (see e.g., [19]).

A d-dimensional grid (or mesh) is a graph whose natural embedding into the Euclidean space $\mathbb{R}^d$ forms a d-dimensional tiling and a path-graph is a one-dimensional grid; torii are "wrap-around" grids and a ring is a one-dimensional torus; and the nodes of a hypercube in an Euclidean-embedding form the corners of a d-cube. These graphs offer multiple min-hop paths between most node pairs and, as it turns out, the way we select from these greatly affects the entropy bounds. Consider, for instance, hypercubes and suppose first that we control next-hop selection. Then, any node can reach half of the nodes on some shortest path via its first port, half of the remaining nodes via its second port, etc., and choosing these shortest paths $H_0 \sim 2$. If, however, next-hop selection is random, then every port "covers" roughly the same number of nodes so $H_0 = I = \log d$.

Accordingly, under the name independent model an optimistic next-hop selection results very small constant zero-order entropy and significant compression, while the pessimistic choice yields no compression at all for these graphs. For

[6]We briefly address weighted graphs in Section VI-A. For BGP-style valley-free policies, the reader is referred to [23].

the name dependent model, we did the analysis only for an optimal choice of forwarding paths (this is in line with the compact routing literature), resulting asymptotically infinite compression[7]. The optimal address space itself is a natural coordinate-wise enlisting of the nodes for the grid and the torii and a lexicographic enumeration on the coordinates for the hypercube. Observe that both strategies tend to keep topologically close nodes close to each other in the address space. The corresponding bound from compact routing is $d\frac{\log n}{n}$, as the 1-interval routing scheme applies [19]. This matches our bounds.

Trees are cycle-free graphs. The respective space bounds depend on the actual tree topology, but the best bounds arise on stars and the worst cases are path-graphs. Still, every tree admits infinite compression over a post-order enumeration of the nodes. The corresponding bounds from compact routing are $5\log n + 1$ bits addresses and $3\log n + \log\log n + 4$ bits forwarding tables in [41] and $(1 + o(1))\log n$ bits addresses and $O(1)$ bits forwarding tables in [11]. These, however, come from specialized non-flat addressing techniques. Strikingly, we get similar results here over flat addresses and without the piecemeal analysis.

Finally, we note that, contrary to what is available in the literature, the above characterizations are tight; the entropy bound is not just a limit on the compressibility of forwarding tables but it also betokens that *no routing scheme can navigate a graph in smaller space*, even if that scheme does not use forwarding tables at all. This shows the power of an information-theoretic approach over a case-by-case analysis.

B. Heuristics on General Graphs

Next, we discuss optimal address space design for general graphs under the name dependent model. As solving (5) directly as a mathematical program seems daunting, we stick to a heuristic approach.

Our heuristic is, by and large, based on hierarchical routing [8]. State aggregation in hierarchical routing occurs using a hierarchical partitioning of the nodes into increasingly smaller groups (or clusters). The top-level cluster contains all nodes and lower-level clusters iteratively partition the respective upper-level cluster into smaller groups. A node's address is the top-down concatenation of cluster ids that contain it. This allows each node to calculate the lowest common cluster to any other node, which will then serve as an index into the forwarding table for sending packets to that node. The forwarding table itself contains one entry per node in the same lowest-level cluster, one entry per each parent-level cluster, etc., all the way to the top-level. In the example of Fig. 2c, node `1.1.1` maintains a separate entry for each node in its own cluster (i.e., all nodes with addresses `1.1.`$\star$), one entry per each second-level cluster (addresses `1.`$\star$), and one for each top level cluster.

Such aggressive state aggregation necessarily incurs path-length increase though; in the example `1.1.1` maintains a single entry for each node whose address matches `2.`$\star$`.`$\star$, even though optimally it would need to use separate ports to reach `2.1.1` and, say, `2.2.3`. Such deviations are exactly why hierarchical routing breaks for optimal routing: the more the underlying topology diverges from a hierarchy the more forwarding rule "exceptions" we would need to store, ballooning forwarding state. And it is precisely such exceptions in a forwarding table what shape its compressibility: the worse the prediction of a node's next-hop from its context (i.e., from the next-hop ports of the nodes mapped to preceding ids) the larger the higher-order entropy and the corresponding space bounds. Yet, *we are still able to show that the general idea of hierarchical routing works reasonably even for optimal routing*: below we define a heuristic clustering scheme and an enumeration method for the nodes inside each cluster, so that the resultant address space will yield significant space reduction under the name independent model, in terms of Theorem 1, in structured graphs.

Recall, the main goal of the partitioning scheme is to map nodes close to one another into the same low-level cluster and distant nodes to separate clusters. Then, a suitable enumeration on the resultant hierarchy (say, a post-order traversal) will readily supply the node ids. Using this observation, we propose a *hierarchical clustering* to design heuristic address spaces for general graphs. Hierarchical clustering is used in machine learning and data mining to heuristically classify nodes reflecting their similarity. Here, we treat nearby nodes "similar" and distant nodes "dissimilar" as of the min-hop distance metric; thus the hierarchical clustering scheme will organize close nodes to the same cluster with high probability; and the subsequent post-order enumeration of the resultant dendogram will heuristically produce a "good" address space for the graph.

The general HIERARCHICAL-CLUSTER scheme implements a simple possible bottom-up classification scheme: it starts with a set of single-node clusters $\mathcal{C} = \cup_v \{v\}$; in each iteration finds the "closest" clusters, where cluster distance is taken by the so called *single linkage rule* $d(C_1, C_2) = \min_{v \in C_1, u \in C_2} d(v, u)$; unifies the two clusters into a single cluster and puts it to the head of $\mathcal{C}$; and goes on with the recursion until $\mathcal{C}$ becomes a single super-cluster containing all of V.

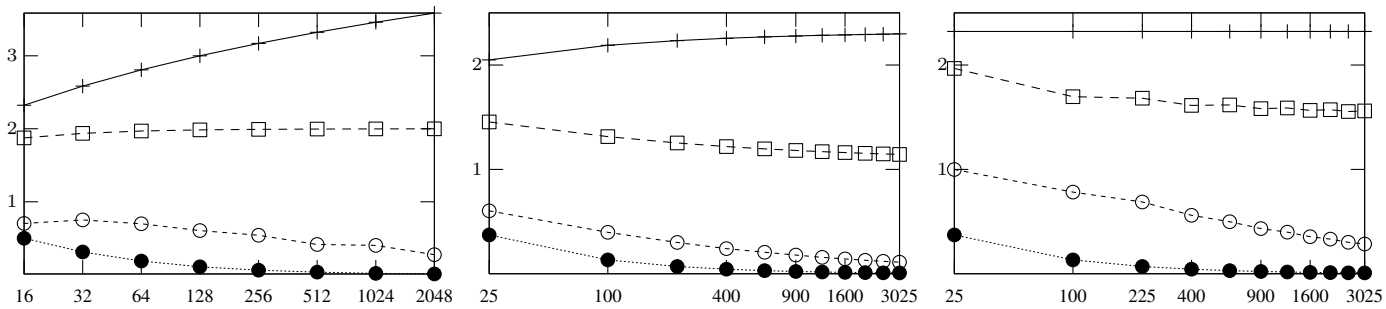
1: **procedure** HIERARCHICAL-CLUSTER($\mathcal{C}$)
2: **if** $|\mathcal{C}| = 1$ **then** STOP
3: $(C_1, C_2) \leftarrow \operatorname{argmin}_{(X,Y) \in \mathcal{C} \times \mathcal{C}} d(X, Y)$
4: $\mathcal{C} \leftarrow (C_1 \cup C_2) \bigcup (\mathcal{C} \setminus C_1, C_2)$

In the below, we use the "fast greedy" hierarchical clustering scheme from [43]; this algorithm runs in $O(n \log n)$ time on a graph of n nodes and, thusly, remains tractable even on very large graphs.

C. Evaluation

Next, we evaluate our address space design heuristic. First, we are interested in the efficiency of the partitioning schemes for producing good address spaces that admit small entropy

[7] $f(n) = o(1)$ means f converges to zero as $n \to \infty$.



(a) hypercube (b) 2-grid (c) 2-torus

Fig. 3: Mean space bounds for various address spaces as the function of graph size n in hypercubes, 2-grids, and 2-torus graphs. Legend: $\bar{I}$ (+), $\bar{H}_0$ (□), $\bar{H}_1^{\text{HC}}$ (○), and $\bar{H}_1^{\text{opt}}$ (●), each result is in bits.

bounds. Thus, initially we study input graphs where the ground-truth *is* known, namely, the well-known graph classes of Section V-A for which we can solve (5) optimally. Then, we turn to general topologies to see how address space optimization affects forwarding table sizes in the wide.

First, we present the analysis over the well-known graphs. The results for the evaluations for min-hop routing on hypercubes, 2-grids, and 2-torus graphs are given in Fig. 3. In particular, $\bar{I}$ denotes the mean information-theoretical lower bound, which is, recall, the best bound we can get in the graph-independent model; $\bar{H}_0$ is the mean zero-order entropy that gives a tight bound in the name independent model, $\bar{H}_1^{\text{HC}}$ is the first-order entropy over the address space calculated by the HIERARCHICAL-CLUSTER heuristic, and finally $\bar{H}_1^{\text{opt}}$ is the theoretical optimum from Table I. In the evaluations when multiple shortest paths were available between two nodes we let the implementation to pick the next-hop port randomly; accordingly the results are conservative estimates (in reality, we could pick the next-hop for each source-destination pair that minimizes the entropy).

Our observations are as follows. First, in the examined cases the zero-order entropy $\bar{H}_0$ was significantly smaller than the information-theoretic limit, suggesting that *substantial forwarding table compression can be achieved even in the name independent model*, without any control over the node addresses. In hypercubes, for instance, $\bar{I}$ grows without limit but $\bar{H}_0 \sim 2$ regardless of the size of the graph, meaning that we pay only 2 bits for every added node. Similar observations hold for grids and torii. Second, *optimal address space design*, and the heuristics discussed in the previous section in particular, are *highly effective in reducing the memory requirement* even below what is already available in the name independent model. For instance, using the hierarchical clustering scheme in an 55×55 grid topology the entire compressed forwarding table for a node takes only 42(!) bytes in average. The heuristic results nicely approach the optimum; all results are in line with the analytic bounds of Table I.

We posit that the positive results are due to that the examined graph topologies are highly symmetric and this allows the address space to reflect the structure that is present in the topology (cf. Rekhter's Law, [1]). To check this proposition, we also ran the evaluations on general graphs that lack such structure. In particular, Table II present the results for some illustrative graph examples that contain varying degrees of internal structure.

TABLE II: Mean space bounds for various address spaces over different graph topologies: number of nodes n and edges m, mean degree $\bar{\delta}$, and the space bounds $\bar{I}$, $\bar{H}_0$, $\bar{H}_1^{\text{rnd}}$, and $\bar{H}_1^{\text{HC}}$ in bits.

Graph	n	m	$\bar{\delta}$	$\bar{I}$	$\bar{H}_0$	$\bar{H}_1^{\text{rnd}}$	$\bar{H}_1^{\text{HC}}$
Tree-2	2000	1999	1.99	0.99	0.0271	0.027	0.0079
Hyper-A	1500	6656	8.87	3.14	0.627	0.612	0.16
Hyper-B	2500	11840	9.472	3.24	0.445	0.4376	0.139
2-core	14056	48722	6.93	2.79	0.4976	0.494	0.421
3-core	7054	34827	9.87	3.30	0.7357	0.726	0.672
4-core	4127	26203	12.69	3.66	1.0041	0.985	0.922
5-core	2603	20240	15.55	3.95	1.291	1.253	1.175
6-core	1742	16024	18.39	4.20	1.5737	1.507	1.424
ER-3	2000	3060	3.06	1.61	1.373	1.369	0.875
ER-6	2000	5995	5.9	2.56	2.19	2.18	2.06
ER-8	2000	7841	7.8	2.96	2.613	2.583	2.5

First, we picked random tree topologies as "maximally structured" examples. Second, we use the hyperbolic large-scale Internet models from [20] where a "hidden" hyperbolic metric space drives network formation and as such, provides reasonable internal structure in the resultant graphs (even though the graphs themselves are random). Third, we examined the Internet inter-domain AS-level topology maps from the CAIDA Autonomous-System-level dataset as of June, 2014 [44]. For this dataset, we took the k-core for different settings of k to obtain a graph ensemble with vanishing internal structure for growing ks. Finally, we studied Erdős-Rényi random graphs, arising as provisioning an edge between each node-pair by an independent Bernoulli-distribution of (given) parameter p. These graphs stand for the "maximally unstructured" case in our evaluations.

The results are for min-hop routing and forwarding tables are sampled at every 50-th node. As far as we know, no analytic space bounds are available for any of these graphs [45]. For address space design, we again used the HIERARCHICAL-CLUSTER heuristic ($\bar{H}_1^{\text{HC}}$), plus we also include the results for a randomly chosen address space as an outlier ($\bar{H}_1^{\text{rnd}}$).

Our results support the above proposition. For highly structured graphs (like general trees or hyperbolic random graphs) *we see substantial space improvement even in the name-independent model*, while *using our engineered address spaces the results for the name-dependent case are even better*. In general trees, in particular, the improvement is in the order-of-magnitude range. *As structure vanishes*, like in the Internet cores for increasing k, *we see less and less routing table compression* under all models (name-dependent, name-independent). At the extreme, on Erdős-Rényi graphs that are completely homogeneous and so, intuitively, one would hardly expect to find a representative hierarchy along which to organize the address space, we see basically no difference in the space bounds obtained with random and optimized address spaces. For large graphs of this family, results gradually approach the worst-case bound from the literature [19], [42].

VI. Applicability

In the next section we show that our results, although stated under stringent initial modeling assumptions, are applicable in a realistic setting as well, even for real-life networks.

First, we show how the results from our flat addressing scheme can be applied to structured address spaces, like the addressing regime the current Internet is based upon. Then, we show that the results are applicable even in realistic weighted networks (most of our evaluations are for the min-hop distance metric). Finally, the static nature of the model is relaxed, by introducing moderate variation in the network structure and examining the effects of adaptive routing.

A. Structured Address Spaces

The most prominent large-scale network of our days is the Internet. Similarly to Kleinrock's hierarchical routing scheme, Internet addressing is also *structured*: host addresses are aggregated into varying sized subnets based common address prefixes and forwarding tables specify routes with respect to those prefixes using the longest-prefix match semantics. Lookup can be implemented with a simple *binary trie* data structure. The information-theoretic model of this paper, however, is for structureless *flat* addresses and *sequential* forwarding tables, which may suggest that our results do not apply to Internet routing. Below, we argue that this is not the case, in that our information-theoretic model gives meaningful memory characterizations even for the structured address space used in IP routing.

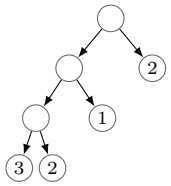
Let T be the binary trie representation of some IP forwarding table in prefix-free form (i.e., after eliminating less specifics [6]), let W be the depth and let n be the number of leaves in T, and let M_T be the minimum space bound

Addr/plen	label
-/0	2
01/2	1
000/3	3

(a)

$s = 32112222$

(c)



(b)

Fig. 4: Representations of an IP forwarding table: (a) tabular form, (b) binary trie for the prefix-free form, (c) equivalent string representation.

for storing T. Furthermore, let S be an equivalent string-representation for T, obtained as concatenating for each individual W-bit address the corresponding next-hop port into a string of 2^W entries (see Fig. 4). Let M_S be the compressed size of S.

Lemma 1. *$M_T \leq M_S \leq KM_T$ for some $1 \leq K \leq 1 + \frac{1}{2}\log\frac{2^W}{n}$.*

See the proof in the Appendix. For a typical IPv4 forwarding table we get $K \sim 3$ and for IPv6 $K \sim 7$. Consequently, the size of T is intimately connected to the size of S and so our compressed string-representation very closely models trie-based IP forwarding tables. The corresponding memory sizes, obtained on real-life IPv4 forwarding tables from [6] with $W = 24$, indicate that the real value of K might be closer to 1.1–1.5.

	M_T [Kbytes]	M_S [Kbytes]	K
taz	56	85	1.52
hbone	142	186	1.3
access(d)	90	118	1.31
as1221	115	162	1.4
as4637	41	66	1.6
as6447	277	294	1.06
as6730	209	253	1.21
fib_600k	157	168	1.07

B. Weighted Graphs

Next, we argue that the above memory requirement characterizations above are conservative in a very strong sense.

Consider the name independent model. We have seen that on a complete graph K_n no forwarding table compression can be achieved: $H_0 = \log n$. The reason is that such a graph is totally homogeneous: each next-hop port appears exactly once in the routing function. We observe, nevertheless, that if the network exhibits some form of heterogeneity then the port id distribution becomes skewed, which tends to reduce entropy. Let us break the symmetry by introducing a slight random edge weight variation and switch to shortest path routing.

Lemma 2. *Let K_n be the complete graph on n nodes and let the weight on each edge be an i.i.d. random variable chosen from $Exp(1)$. Then, for a randomly picked node v: $\mathbb{E}(H_0(v)) = \log e$ as n grows to infinity.*

The proof is based on that the size of the branches of the shortest path tree can be described by the Chinese restaurant

TABLE III: Results for the dynamic setting: the effects of randomly removing edges from the graph instance Hyper-B.

Removed edges %	$\bar{H}_0$	$\bar{H}_1^{\text{HC}}$
0	0.360	0.137
1	0.360	0.139
2	0.365	0.145
5	0.362	0.163
10	0.389	0.188
20	0.517	0.261

model [46]. Note that the result remains true for a wide variety of i.i.d. weight distributions. What is remarkable in this finding is that just the slight diversity introduced by random weights is enough to reduce the entropy from $\log n$ to $\log e \approx 1.44$ bits. This is not even dependent on the network size, even though the number of ports grows without limit.

C. Dynamic Networks and Adaptive Routing

Finally, we argue that our models and address space design methods are applicable even in a dynamic setting.

The above analytical and empirical results are strictly for the case when routing tables remain static. However, real world networks tend to change over time. When the graph topology changes, e.g., a link is removed from the graph due to a link failure, some next-hop entries at some nodes also change and, consequently, the routing tables, and the underlying address space that was designed for the *original* graph, may shift out of alignment. This may result in substantial increase of compressed routing table size due to dynamic routing.

Below, we argue that this is not the case. First, we take the same hyperbolic graph instance of 2500 nodes (Hyper-B) we examined previously (see Table II, [20]) and we design an initial address space using the HIERARCHICAL-CLUSTER heuristic. Then, we randomly remove edges from the graph and we observe whether, and to what extent, routing table entropy $\bar{H}_1^{\text{HC}}$ changes, when the entropy is taken over the initial address space that was designed for the *original* graph (ideally, the address space should be re-designed after every graph alteration). This will allow us to assess whether a heuristic address space is *robust* to dynamism.

The results are presented in Table III. We observe that randomly removing 1–2% of the edges has negligible effect on the first-order routing table entropy $\bar{H}_1^{\text{HC}}$. This suggests that *optimal address spaces are oblivious to slow-pace changes to the graph topology*. Even if 10% of the edges is randomly removed we see only roughly 10% increase in the entropy. (Recall, this is without re-designing the address space for the altered graphs.) However, removing every fifth edge ultimately doubles the entropy, but even in that case the entropy for the initial address space ($\bar{H}_1^{\text{HC}}$) remains significantly smaller than that for the unoptimized case ($\bar{H}_0$).

VII. Conclusions

The size and the rate of growth of Internet forwarding routing tables has raised considerable interest recently [1], [5]. Even though growth in itself does not necessarily indicate a grave scaling problem [4], the continuing expansion of the memory footprint of Internet packet forwarding has put ample stress on the operational network infrastructure and to a large extent questioned the soundness of the underlying design principles [2], [3].

Curiously, we found that the effective information content Internet forwarding tables actually store has not increased that dramatically. Just the contrary, forwarding table entropy has remained remarkably low and relatively stable over the last 18 months. Motivated by the demand to uncover the systematic reasons behind this finding, in this paper we presented a principled, information-theoretic analysis of the memory requirement of packet routing in growing networks.

We described forwarding tables as simple sequential strings and we used the conventional notion of Shannon-entropy to characterize the respective memory requirements. This simple model seems just sufficiently and necessarily rich, omitting the uninteresting subtleties while still delivering meaningful answers. We could subject the model to large-scale numeric evaluations as well as to mathematical analysis, and the emergent space bounds match perfectly the corresponding results available in the compact routing literature, without the piecemeal analysis. The model could even be implemented in routers, see e.g. [6], [28], [30], [31] on how to realize fast lookups on compressed sequential forwarding tables.

Still, the most important virtue of the model in our view is that it allowed us to give a completely new interpretation of "address spaces", a central concept in networking. Namely, using the notion of higher-order entropies we could for the first time reason quantitatively and qualitatively on the extent of correlation between node addresses and the underlying topology. So far, such argumentation has been largely intuitive. We used this observation to formulate the optimal address space design problem as the task to set node ids to minimize the mean k-order routing entropy and we gave precise solutions on highly symmetric graphs and heuristic characterizations for the general case. Strikingly, our results consistently indicate space reductions even on random node addresses, over which our address space designs normally improve further considerably.

Of course, this paper means just the first step to understand the intricate connections between network structure, address spaces, and forwarding state entropy. Yet, even in this early phase of the research we can safely say that the Internet may not be an isolated example of a network with low and stable routing entropy. Contrarily, basically all but the most artificial examples (e.g., Erdős-Rényi random graphs) exhibit similar phenomena. This finding seems to substantiate our belief that address space optimization coupled with forwarding table compression can be a potentially useful tool in scaling the hop-by-hop routing paradigm into the future.

Acknowledgements

Project nos. 123957, 129589 and 124171 have been implemented with the support provided by the National Research, Development and Innovation Fund of Hungary, financed under

the FK_17, KH_18 and K_17 funding schemes respectively. The research reported in this paper has also been supported by the National Research, Development and Innovation Fund (TUDFO/51757/2019-ITM, Thematic Excellence Program). Z. Heszberger has been supported by the Janos Bolyai Fellowship of the Hungarian Academy of Sciences and by the UNKP-19-4 New National Excellence Program of the Ministry of Human Capacities.

REFERENCES

[1] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984, 2007.

[2] Xiaoliang Zhao, Dante J. Pacella, and Jason Schiller. Routing scalability: an operator's view. *IEEE JSAC*, 28(8):1262–1270, 2010.

[3] Varun Khare, Dan Jen, Xin Zhao, Yaoqing Liu, Dan Massey, Lan Wang, Beichuan Zhang, and Lixia Zhang. Evolution towards global routing scalability. *IEEE JSAC*, 28(8):1363–1375, 2010.

[4] Kevin Fall, Gianluca Iannaccone, Sylvia Ratnasamy, and P. Brighten Godfrey. Routing tables: Is smaller really much better? In *ACM HotNets-VIII*, 2009.

[5] The Register. The internet just broke under its own weight–we explain how. http://www.theregister.co.uk/2014/08/13/512k_invited_us_out_to_play, 2014.

[6] Gábor Rétvári, János Tapolcai, Attila Kőrösi, András Majdán, and Zalán Heszberger. Compressing IP forwarding tables: towards entropy bounds and beyond. In *ACM SIGCOMM 2013*, pages 111–122, 2013. Revised version: http://arxiv.org/abs/1402.1194.

[7] A. Kőrösi, J. Tapolcai, B. Mihálka, G. Mészáros, and G. Rétvári. Compressing IP forwarding tables: Realizing information-theoretical space bounds and fast lookups simultaneously. In *IEEE ICNP*, 2014.

[8] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks, performance evaluation and optimization. *Computer Networks*, 1(3):155–174, 1977.

[9] J. McQuillan. Adaptive routing algorithms for distributed computer networks. BBN Rep. 2831, Bolt Beranek and Newman Inc., 1974.

[10] Dmitri Krioukov, kc claffy, Kevin Fall, and Arthur Brady. On compact routing for the Internet. *SIGCOMM Comput. Commun. Rev.*, 37(3):41–52, 2007.

[11] M. Thorup and U. Zwick. Compact routing schemes. In *ACM SPAA'01*, pages 1–10, 2001.

[12] J. J. Aceves-Luna-Garcia. Routing management in very large-scale networks. *Future Gener. Comput. Syst.*, 4(2):81–93, 1988.

[13] M. O'Dell. GSE – an alternate addressing architecture for IPv6. Internet-draft, IETF, 1997.

[14] I. Castineyra, N. Chiappa, and M. Steenstrup. The Nimrod routing architecture. RFC 1992, 1996.

[15] F. Kastenholz. ISLAY: A new routing and addressing architecture. Internet-draft, IETF, 2006.

[16] Feng Wang, Lixin Gao, Xiaozhe Shai, Hiroaki Harai, and Kenji Fujikawa. Compact location encoding for scalable internet routing. In *INFOCOM*, 2015.

[17] A. Gulyas, G. Retvari, Z. Heszberger, and R. Agarwal. On the scalability of routing with policies. *Networking, IEEE/ACM Transactions on*, PP(99):1–1, 2014.

[18] Tong Yang et al. Constant IP lookup with FIB explosion. *IEEE/ACM Trans. Netw.*, 26(4):1821–1836, 2018.

[19] Cyril Gavoille and Stéphane Pérennès. Memory requirement for routing in distributed networks. In *ACM PODC*, pages 125–133, 1996.

[20] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

[21] Dmitri Krioukov, Kevin Fall, and Xiaowei Yang. Compact routing on internet-like graphs. In *IEEE INFOCOM 2004*, volume 1. IEEE, 2004.

[22] Marián Boguná, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature Communications*, 1:62, 2010.

[23] G. Rétvári, D. Szabó, A. Gulyás, A. Kőrösi, and J. Tapolcai. An information-theoretic approach to routing scalability. In *ACM HotNets-VIII*, pages 1–7, 2014.

[24] Paolo Ferragina, Fabrizio Luccio, Giovanni Manzini, and S. Muthukrishnan. Compressing and indexing labeled trees, with applications. *J. ACM*, 57(1):1–33, 2009.

[25] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, 1991.

[26] Darrel Hankerson, Peter D. Johnson, and Greg A. Harris. *Introduction to Information Theory and Data Compression*. CRC Press, Inc., 1998.

[27] Statistical distributions of english text. http://www.data-compression.com/english.shtml.

[28] Paolo Ferragina and Rossano Venturini. A simple storage scheme for strings achieving entropy bounds. In *ACM-SIAM SODA*, pages 690–696, 2007.

[29] Giovanni Manzini. An analysis of the Burrows–Wheeler Transform. *J. ACM*, 48(3):407–430, 2001.

[30] Paolo Ferragina, Raffaele Giancarlo, and Giovanni Manzini. The myriad virtues of wavelet trees. *Inf. Comput.*, 207(8):849–866, 2009.

[31] Jérémy Barbay, Meng He, J. Ian Munro, and Srinivasa Rao Satti. Succinct indexes for strings, binary relations and multilabeled trees. *ACM Trans. Algorithms*, 7(4):52:1–52:27, 2011.

[32] Veli Mäkinen and Gonzalo Navarro. Dynamic entropy compressed sequences and full-text indexes. *ACM Trans. Algorithms*, 4(3):32:1–32:38, 2008.

[33] Paolo Ferragina, Rodrigo González, Gonzalo Navarro, and Rossano Venturini. Compressed text indexes: From theory to practice. *J. Exp. Algorithmics*, 13:12–31, 2009.

[34] Gonzalo Navarro and Veli Mäkinen. Compressed full-text indexes. *ACM Comput. Surv.*, 39(1), 2007.

[35] Wing-Kai Hon, Rahul Shah, and Jeffrey Scott Vitter. Compression, indexing, and retrieval for massive string data. In *CPM*, pages 260–274, 2010.

[36] Nivio Ziviani, Edleno Silva de Moura, Gonzalo Navarro, and Ricardo Baeza-Yates. Compression: A key for next-generation text retrieval systems. *IEEE Computer*, 33(11):37–44, 2000.

[37] Marta Arias, Lenore J. Cowen, Kofi A. Laing, Rajmohan Rajaraman, and Orjeta Taka. Compact routing with name independence. In *ACM SPAA*, pages 184–192, 2003.

[38] João Luís Sobrinho. An algebraic theory of dynamic network routing. *IEEE/ACM Trans. Netw.*, 13(5):1160–1173, 2005.

[39] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. *ACM Trans. Algorithms*, 4(3):37:1–37:12, 2008.

[40] J. Van Leeuwen and R. B. Tan. Interval routing. *Comput. J.*, 30(4):298–307, 1987.

[41] P. Fraigniaud and C. Gavoille. Routing in trees. In *ICALP '01*, pages 757–772, 2001.

[42] Cyril Gavoille and Marc Gengler. Space-efficiency for routing schemes of stretch factor three. *J. Parallel Distrib. Comput.*, 61(5):679–687, 2001.

[43] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[44] The CAIDA AS relationships dataset (2013-08-01). http://www.caida.org/data/active/as-relationships.

[45] Mihaela Enachescu, Mei Wang, and Ashish Goel. Reducing maximum stretch in compact routing. In *INFOCOM*, pages 336–340, 2008.

[46] Svante Janson. One, two and three times log n/n for paths in a complete graph with random weights. *Comb. Probab. Comput.*, 8(4):347–361, July 1999.

VIII. Appendix

Proof of Lemma 1: Ref.[6] shows that $M_T = n(2 + H_0^T)$, where H_0^T is the zero-order entropy of the empirical next-hop port distribution on the leaves of T. Since M_T is minimal $M_T \leq M_S$. Next, we show $M_S \leq KM_T$ for proper K.

Let $\frac{n_i}{n} : i \in [1, \delta]$ and $\frac{n_{ij}}{n} : i, j \in [1, \delta]$ be the zero- and first-order statistics of the empirical port distribution on the leaves of T and let $\frac{l_i}{2^W} : i \in [1, \delta]$ and $\frac{l_{ij}}{2^W} : i, j \in [1, \delta]$ be the zero- and first-order statistics of the empirical symbol distribution in S. Denote by H_1^S the first-order empirical entropy of S. Since $\frac{1}{x} \log x$ is concave:

$$H_1^S = \sum_i \frac{l_i}{2^W} \sum_j \frac{l_{ij}}{l_i} \log \frac{l_i}{l_{ij}}$$
$$\leq \sum \frac{l_i}{2^W} \left(\frac{l_i^*}{l_i} \log \frac{l_i}{l_{i*}} + \sum \frac{l_{ij}}{l_i} \log \frac{l_i}{l_{ij}} \right) \ ,$$

where l_i^* is the number of times port i is followed by port i mapped from the same leaf of T and l_{ii} is when the second i comes from another leaf. Then, $l_{ij} = n_{ij}$ and $l_i^* = l_i - n_i$ and hence

$$2^W H_1^S \leq \sum_i (l_i - n_i) \log \frac{l_i}{l_i - n_i} + \sum_i n_i \sum_j \frac{n_{ij}}{n_i} \log \frac{n_i}{n_{ij}}$$
$$\leq n \log e + n \sum_i \frac{n_i}{n} \log \frac{l_i}{n_i} + nH_1^T \ ,$$

where $H_1^T \leq H_0^T$ is the first-order entropy of the leaf-string in T. We get that $M_S = 2^W H_1^S \leq n(\log e + (W - \log n) + H_0^T)$ and hence $K = \frac{M_S}{M_T} \leq \frac{H_0 + \log e + W - \log n}{H_0 + 2}$. For IPv4 in particular, $H_0 = O(1)$ and $n \sim 150,000$ [6] and so taking $W = 24$ we get $K \sim 3$. Similar argumentation for IPv6 yields $K \sim 7$. ■