Budapest University of Technology and Economics (BME)
Faculty of Electrical Engineering and Informatics (VIK)
Department of Telecommunications and Media Informatics (TMIT)
High-Speed Networks Laboratory (HSN*Lab*)
MTA-BME Information Systems Research Group

Interdisciplinary Approaches to Open Problems in Network Communications

D.Sc. Dissertation

In partial fulfillment of the requirements for the title of
Doctor of the Hungarian Academy of Sciences

Gábor Rétvári, Ph.D.

Magyar tudósok körútja 2.
H-1117 Budapest, Hungary,
E-mail: retvari@tmit.bme.hu

Budapest
2020

Dedication

This Thesis is dedicated to my Wife, my Mother, and my Family.

Acknowledgments

This work was carried out at the Department of Telecommunications and Media Informatics (TMIT), Budapest University of Technology and Economics (BME) during the years 2007—2020. I am grateful for the support of the High-Speed Networks Laboratory (HSN*Lab*), the MTA–BME Information Systems Research Group, the MTA–BME Momentum Future Internet Research Group, the MTA–BME Momentum Network Softwarization Research Group, the MTA János Bolyai Research Fellowship, the NKFIH–OTKA Postdoctoral Excellence Programme, and Ericsson Research, Hungary.

My warmest thanks are due to my closest co-authors and colleagues, Felicián Németh, János Tapolcai, András Gulyás, Zalán Heszberger, Balázs Sonkoly, László Toka, and József Bíró, my former and present PhD students, Gábor Enyedi, Levente Csikor, Krisztián Németh, Máté Nagy, Gábor Németh, and Tamás Lévai, and fellow researchers from Ericcson TrafficLab, András Császár, László Molnár, Gergely Pongrácz, and Szabolcs Malomsoky. Special thanks go to Attila Kőrösi for his endless tolerance of my lack of mathematical skills. I am grateful to all colleagues of the Lab and the Department for the inspiring atmosphere.

I wish to express my gratitude to my international collaborators Marco Chiesa (KTH), Michael Schapira (HUJI), Stefan Schmid (Univ. of Vienna), Barath Raghavan (USC), and Gianni Antichi (Queen Mary University). I am especially honored by the invitations that allowed me to visit some of the greatest universities and most recognized researchers of the world: Sylvia Ratnasamy and Scott Shenker at UC Berkeley, Jennifer Rexford at Princeton, Michael Schapira at the HUJI, and Ori Rottenstreich at the Technion. Special thanks are due to Sergey Gorinsky (IMDEA Networks) for helping me find my way in the systems community.

Last but not least, I wish to thank my wife Marcsi, for her love and humor that got me through lots of difficulties in life, my Mother for her patience during my endless procrastication before finishing this Dissertation, and my brother László, her wife Zsófi, and their two fantastic sons, Ádám and Bálint, for all the fun we had together.

Project no. 104939, 108947, 123957, 124171, and 129589 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the PD_12, K_13, FK_17, K_17, and KH_18 funding schemes, respectively. Gábor Rétvári was supported by the János Bolyai Fellowship of the Hungarian Academy of Sciences.

Contents

Chapter 1

Preface

The grand challenges facing society—energy, water, climate, food, health—have become too complex to be tacked within any single academic discipline alone. With academics, research institutes, and government funding agencies increasingly inciting scientists to break traditional disciplinary silos and bring multiple diverse academic research fields together to deal with humanity's greatest problems, interdisciplinary research has become mainstream over the last decades.

1.1 Interdisciplinarity

Scientific advancement was traditionally made within well-defined scientific disciplines, e.g. physics, chemistry, and biology. Loosely speaking, a discipline is a community of scientists agreeing on a common set of challenges, terminology, methodology, expertise, and practices, i.e., a scientific paradigm [128], strongly associated with a given scholastic subject area. The reductionist approach facilitated significant scientific advancement in the last centuries, but at the same time divided the scientific community into isolated groups of specialists living in their walled gardens, developing their own language, forums, and paradigm, pursuing minimal interaction with other scientific communities.

Modern societal problems, however, transcend conventional academic boundaries. Consequently, there is a growing need for disciplines to collaborate in order to create something more than the sum of their parts, without being constrained by one way of thinking or tackling a problem. *Interdisciplinary research* targets such overarching research problems, combining skills and knowledge from a variety of disciplines in a scientific process that is much more integrated and efficient than working in groups divided by subject.

The definition of what constitutes a "discipline" and what defines "interdisciplinarity" has occupied much scholarly debate. Below, we embrace the following interpretation [161]:

> *"Interdisciplinary research is a mode of research by teams or individuals that integrates information, data, techniques, tools, perspectives, concepts, and/or theories from two or more disciplines or bodies of specialized knowledge to advance fundamental understanding or to solve problems whose solutions are beyond the scope of a single discipline or area of research practice."*

Interdisciplinarity has broad vocabulary [198]: broadly speaking, we distinguish traditional *intradisciplinary* research that is working within a single discipline, *crossdisci-*

plinary research that views one discipline from the perspective of another, *multidisciplinary* research where a select group of people from different disciplines works together, each drawing on their disciplinary knowledge, and *interdisciplinary* or *transdisciplinary* research where the emphasis is on integrating knowledge and methods in order to synthesize a unity of intellectual frameworks beyond the disciplinary perspectives.

The promise of interdisciplinary research is that one can tap into the expertise of a wide range of disciplines, acquire a wider academic perspective, and learn the thinking styles of other disciplines. This *interdisciplinary approach often yields new insights, unexpected results, far-fetching consequences*, and spawns entirely new disciplines. Nevertheless, moving into a new discipline and culture will present challenges, too. Most importantly, interdisciplinary research requires specialists to familiarize themselves with the new theory, methodology, and practice of the subjects they are moving into. Further challenges are mostly cultural in nature: every discipline has its own jargon and interdisciplinary research requires scientists to understand and learn the new terminology; the current structure of academic departments, including research programs, faculty, staff, and organization, is specialized to a single field, which may make interdisciplinarity collaboration a challenge when the structures across departments do not align; and grant calls, publication venues, and scientific forums are likewise structured around disciplines, making it difficult to secure funding and publish interdisciplinary research results. Accordingly, *interdisciplinary collaboration is the exception in large-scale academic research today rather than the rule.*

This is especially so in the area of computer science and network communications, where the specialized nature of the related engineering fields has produced an immense number of fragmented disciplines and sub-disciplines. As of 2020, the *Association for Computing Machinery* (ACM), the largest international scientific community dedicated to computing, counts 37 special interest groups (SIGs), each devoted to a distinct field of computer science, ranging from general disciplines like computer communications (*SIGCOMM*) and operating systems (*SIGOPS*), to specialized "niche" fields like symbolic & algebraic manipulation (*SIGSAM*) or university and college computing services (*SIGUCCS*). None of these SIGs have interdisciplinary research among its primary goals. Even the individual disciplines are broken into their own respective subfields; within the field of networking the *IEEE Communications Society* (ComSoc) offers 15 different transactions and journals, each specializing in a distinct smaller area within communications, like cognitive communications, green communications, optical networking, or molecular and biological communications. By the best of our knowledge, within the communications discipline there are only two journals, the prestigious *IEEE/ACM Transactions on Networking* and the *IEEE Selected Areas of Communications*, which openly welcome interdisciplinary research on networking.[1]

1.2 Contributions

With the dramatic recent growth of the global Internet, which today connects tens of thousands of autonomous network domains each operated by independent governmental, academic, military, and private enterprise stakeholders, millions of fixed and mobile devices and billions users, and delivers trillions of US dollars in business value, the scale

[1]The author has published 6 papers in the *IEEE/ACM Transactions on Networking* [42, 88, 123, 152, 172, 181] and 2 in the *IEEE Selected Areas of Communications* [136, 204] during the last 15 years, mostly dedicated to interdisciplinary studies.

and complexity of the related theoretical, engineering, societal, and economic challenges has greatly surpassed the capability of individual network communications disciplines to tackle [103]. This calls for an interdisciplinary approach in computer networking, which combines and synthesizes the models, methodologies, and algorithmic toolsets of different disciplines, including control theory, information theory, graph theory and numerical optimization, into a single framework that can be used to attack notorious open problems that have so far eluded the capability of individual disciplines to successfully solve.

The goal of this Dissertation is to foster interdisciplinarity in the field of computer networks and communications, by *presenting a selection of recent advances on related open problems that were made possible by an interdisciplinary research methodology*. A common theme in the reviewed studies is that the interdisciplinary approach was applied in a comprehensive manner, from the initial phase of model formulation and problem statement through the development of the corresponding theoretical frameworks, algorithms and data structures, all the way to the eventual implementations and evaluations, mixing the terminology, methodology, tools and software libraries, from at least two disciplines. This approach then yielded new models that produced answers to several compelling open problems, and new algorithmic tools to experiment with the solutions developed.

1.3 Structure of this Dissertation

Chapter 2 is dedicated to an interdisciplinary study of *fair resource allocation problems* arising in the field of communication networks. Such resource allocation problems manifest themselves naturally in several disciplines independently; e.g., in economy as the distribution of income and wealth among individuals via markets or planning, in game theory as multi-agent competitions to possess some valuable goods, or in computer networks where multiple users bid for the limited transmission capacity available in a communication network. This inherently multi-disciplinary nature of the resource allocation problem expressly calls for an interdisciplinary approach. In the specific study we present first, the task is to distribute transmission rates among users in a way so that the allocation is feasible, in that the limited capacity transmission links in the network do not become overloaded, and fair, in that none of the users remain discontented with their resource share. The interdisciplinary nature of the approach stands in that (1) the notion of fairness is generalized from the "usual" setting of max-min fairness to other notions of fairness (Pareto-optimality, non-dominatedness, etc.) and (2) a unique *geometric* model is developed that transforms the problem from the conventional flow-theoretical framework to the language of convex geometry. As the most important contribution, we answer the decade-old open problem whether max-min fair allocations can exist in the case when we do not fix the paths of users beforehand [133, Section "When bottleneck and water-filling become less obvious"]. We stress that achieving this result was made possible thanks to the interdisciplinary approach and the use of geometric insights to a problem that is conventionally analyzed in the context of a different discipline.

Chapter 3 presents a recent interdisciplinary approach to *multipath rate-adaptive routing* problems. An adaptive routing algorithm controls the rate at which traffic is routed to each individual forwarding path in the network, in concert with the actual user demands. Again, the emphasis is on feasibility; i.e., avoiding the over-subscription of the limited transmission capacity of the network to avoid congestion. The research presented casts the problem of rate-adaptive multipath routing, which is conventionally approached using the toolset of numerical optimization and flow theory, in the setting of *control the-*

ory. Again, the interdisciplinary approach is enforced right from the modeling phase: the routing problem is formulated in the framework of constrained optimal control theory and an optimal state-feedback routing controller is obtained that is then theoretically proved to be stable, optimal, and feasible. As far as we know, this is the first time that the existence of such an offline multipath routing algorithm is proved, which can route *any* admissible traffic matrix in the network without prior knowledge on the user demands. Again, this is made possible by the interdisciplinary approach, by applying the theory and algorithms of constrained receding horizon control to the multipath routing problem.

Chapter 4 is devoted to an interdisciplinary study of *scalable policy routing* algorithms for large-scale communication networks. In this study, the main concern is to generalize the well-established scalable routing theories from the context of shortest-path-routing to arbitrary path-selection policies that favor a broader set of attributes and descriptors beyond path length (e.g., widest-path routing, secure routing, BGP valley-free routing). Scalability in this setting means that the amount of information that needs to be stored at network nodes does not grow prohibitively with the size of the network. This problem has become especially compelling lately, with the unprecedented growth of the Internet and the rising scalability concerns, given that the global Internet is *not* running on shortest-path routing as prior work presumes. The interdisciplinary approach presented in this Chapter is perhaps the most unorthodox one in this Dissertation; namely, the specifics of routing policies are described using the formalism of *abstract algebra*, which makes it possible to obtain a *generic* understanding of the scalability properties of different routing policies, stated purely in terms of abstract algebraic properties, that goes beyond the piecemeal analyses available in the literature. The most important results here are (1) the comprehensive scalability characterization of most known intra-domain routing policies and the extension of the well-known negative results beyond shortest-path routing; (2) using a novel algebraic generalization of the notion of stretch, several scalable "approximate" routing algorithms for notoriously difficult routing problems that are known to scale poorly in the optimal setting; and (3) the first proof for the existence of certain pathological routing policies for which no scalable realizations exist even if permitting arbitrary constant stretch.

Finally, Chapter 5 is dedicated to the problem of *forwarding table compression*, with the goal to reduce the amount of routing state stored in the nodes of communication networks. This study is closely related to the previous Chapter; whereas the foregoing analysis was deliberately of worst-case nature, considering hypothetical routing algorithms that would provide scalable routing state in *any* network topology, the present study considers the attainable smallest routing state on *particular inputs*. Information-theory and data compression theory lend themselves naturally in this context; surprisingly, the study covered in this Chapter is the first one to cast the problem of forwarding table compression in an information-theoretical framework. The major new result is the fixture of an *entropy* notion to characterize the maximum compression that can be attained on particular forwarding table instances and the definition of several forwarding table compression schemes that, according to the evaluations that we also briefly cover, attain orders of magnitude space reduction beyond the state-of-the-art. Crucially, our encoding schemes are such that they allow to execute fast queries to the compressed forwarding tables without explicit decompression, which makes them especially appealing to practice.

The organization of the subsequent chapters follow the same structure. In each Chapter we provide the general background on the main problem first and we point at the

potential to apply an interdisciplinary approach for the specific problem domain. Next, we present formal models and problem formulations arising in the context of the chapter. Then we turn to describe the main contributions, and finally we review related research and position the new results in the grand theme of the field. Each chapter stands on its own and can be read independently from the rest; whenever there is overlap between the notations in two or more chapters we explicitly point the reader to the full definitions.

Chapter 2

Fairness in Internet Routing: The Geometric Perspective

This Chapter is devoted to the problem of allocating scarce resources in a network so that every user gets a fair share, for some reasonable definition of fairness. For example, a fair allocation would be such that every user gets the same share and the allocation is maximal in the sense that there does not exist any larger, even and feasible allocation. We shall focus on the type of allocation problems that arises most often in networking: allocate a fair traffic rate for each user in a network whose links are of limited capacity (see Fig. 2.1).

2.1 Preliminaries

2.1.1 Fair Resource Allocation in Networks

It is an imminent incentive of the network operator to share the limited resources available in the network, like costly compute nodes, scarce storage facilities, or constrained interconnection capacities, between users in an efficient yet fair manner [26, 107, 115, 132, 169, 171]. Think of, for instance, a multi-tenant public cloud where server space is limited by the physical dimensions of the data center facility [53,81,82,109,110,230], or a transport network where users compete for the bandwidth of long-haul links [24, 57, 115, 189, 201]. If certain users are overflown with resources while others are starved, unfairly traded users may move to alternative operators with a more reasonable resource allocation regime in place. In this context, a fair allocation means a strategy to distribute common wealth in a way that maximizes users' satisfaction with the share of resources they receive.

Among the many different definitions of fairness perhaps the most prevailing one is max-min fairness. A max-min fair allocation is, roughly speaking, such that we cannot increase the share of any of the users without decreasing the share of some other user that already receives less or equal rate [107]. Max-min fairness is a simple yet powerful fairness criterion, and consequently it has grown to be an essential ingredient in diverse fields of networking, like flow and congestion control protocols [24, 53, 57, 92, 189, 201], online job scheduling [81, 82], bandwidth sharing in ATM networks [47, 115], or distributing compute [37, 108, 200], storage [13, 110], memory [4], and network resources [57, 81, 82, 109, 115, 201, 230] in a multi-user computer system, a public cloud, or the Internet; for comprehensive surveys see [26], [132], [58], [110], and [131].

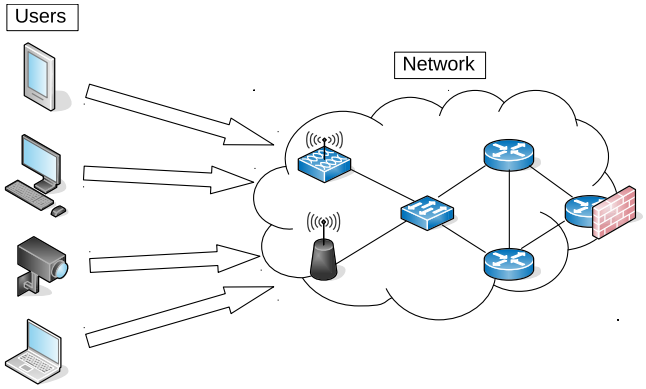


Figure 2.1: Fair resource allocation in capacitated networks: users compete for the scarce resources of the network and the task of the network operator is to arbitrate resources in a way that the network is not over-committed beyond its capacity and each user is satisfied with their share. For instance, an egalitarian resource allocation regime would share resources equally between competing users.

Max-min fairness is most easily described in a network model where a single path is assigned to each user and this path remains fixed during the lifetime of the network. Here, the task is to compute a rate at which users can send data to their path, so that the allocation is max-min fair and neither of the edges gets overloaded. A very useful tool to solve this problem is the notion of bottlenecks [26]: Given any user, the bottleneck edge for this user is an edge with the properties that

(i) it is filled to capacity, that is, users send just enough load to the link so that the bandwidth is fully utilized, and

(ii) the user has the maximum rate amongst the users whose path traverses the edge.

Bottlenecks are very tightly coupled with max-min fairness, for it can be shown that an allocation of rates is max-min fair over some fixed single-path routing if and only if all users have a bottleneck edge.

From the practical standpoint, the importance of this *bottleneck argumentation* is multi-faceted. First, as the name suggests, bottlenecks point to certain shortages of resources in a network that, given the selected set of paths, constrain the fair allocation. Additionally, bottlenecks substantiate a fast algorithm, the so called *water-filling algorithm*, to find a max-min fair allocation [26] (see later).

Curiously, *the actual assignment of paths to users influences the emergent max-min fair allocation to a great extent*, in that different selections of paths will yield different max-min fair allocations. In the conventional approach to bandwidth allocation problems, however, the *forwarding path assigned to users is fixed* and the fair allocation is to be found with regards to this fixed set of paths. This feels arbitrary and unintuitive; after all, it is the specifics of the network, in particular the network topology and link capacities, that fundamentally determine the share of resources that can be allocated to users and not some random routing decision. Accordingly, *we should first compute a max-min fair allocation that is only dependent on the network itself*, and only after this we should pick a routing that realizes it. Below, we shall refer to this problem as *the general max-min fair bandwidth allocation problem* [169–171,173], and all the former incarnations will be called *fixed-path max-min fair bandwidth allocation problems*. The main focus in this Chapter is this generalized version of the bandwidth allocation problem.

2.1.2 Contributions

In the past several attempts have been made to address the general max-min bandwidth fair allocation problem [150], [133]. These works provide a *quantitative* treatment, establishing the existence and uniqueness of a max-min fair allocation in the generalized model and providing several algorithms to compute it, heavily relying on techniques from lexicographical optimization. Below, we *complement existing quantitative arguments with a qualitative treatment*, building a completely new framework using mainly geometric principles [169–171, 173]. Our results reveal the intricate relationship between the specifics of a network and the generalized max-min fair allocation; in particular, *we answer the old problem* (first raised in [133, Section "When bottleneck and water-filling become less obvious"]) *whether the bottleneck argumentation generalizes from the fixed-path model to the routing-independent generic model in the affirmative.*

Our contributions in particular are as follows [169–171, 173].

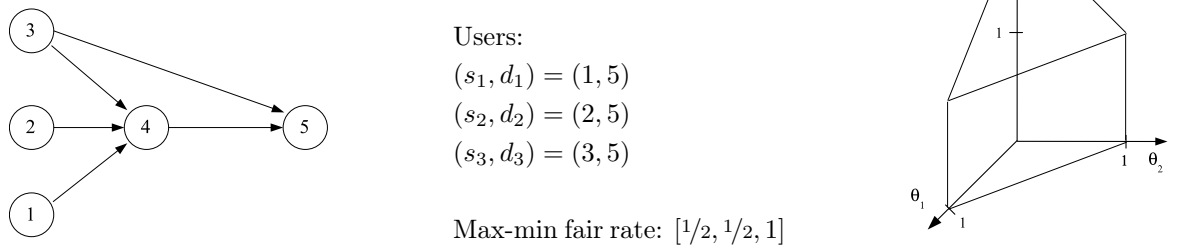
- We show that the set of feasible bandwidth allocations in a capacitated network forms a convex polytope. This result is essential as it then fixes the existence of a max-min fair allocation in the general, routing-independent setting.
- We provide a "geometric" bottleneck argumentation for the general setting, whereby users' bottlenecks are represented as certain supporting hyperplanes of the above feasible set, and we show that the properties for of "fixed-path" bottlenecks naturally extend to our geometric interpretation.
- We translate the geometric notion of bottlenecks back to a graph-theoretical setting that is better suited for operators to reason about resource scarcities in their network; the new interpretation represents bottlenecks as critical cuts in the network which again possess the distinctive properties of "conventional" bottlenecks.

The rest of this Chapter is structured as follows. In Section 2.2 we introduce a model for the general max-min fair bandwidth allocation problem and we give some examples. Then, in Section 2.3 we characterize the set of feasible bandwidth allocations, we use this characterization to (re-)state the existence of generally max-min fair allocations, and we give the geometric and the graph-theoretical bottleneck argumentations. Finally, in Section 2.5 we summarize the related work and position our results in the considerable body of literature on fair rate allocation problems.

2.2 Formal Model

In this Chapter, the task we consider is to compute a transmission rate (or throughput, for short) for each user that is *feasible*, so it can be routed in the network without violating the edge capacities, *efficient*, so that the resources of the network are fully utilized, and, last but not least, satisfies some *fairness* criteria. Perhaps the most instrumental way to understand the context of such fair allocations is through an example.

Example 2.1. Consider the simple directed network of Fig. 2.2a and suppose that there are 3 source-destination pairs (or users or commodities): $(1, 5)$, $(2, 5)$ and $(3, 5)$ (see Fig. 2.2b). All edge capacities are uniformly 1 unit. For example, if we were to arbitrate resources between users strictly evenly then we would allocate $1/2$ transmission rate for each user; say, we could let user $(1, 5)$ to use the path $1 \to 4 \to 5$ and user $(2, 5)$ the path $2 \to 4 \to 5$, sharing the bottleneck resource of capacity 1 on link $(4, 5)$, and user $(3, 5)$ would receive exclusive access to link $(3, 5)$, sending all of its share, $1/2$ units of traffic, to



(a) topology (b) parameters (c) feasible allocations

Figure 2.2: A sample network and the set of rates realizable in it. All edge capacities are equal to 1. There are 3 source-destination pairs $(1, 5)$, $(2, 5)$ and $(3, 5)$, whose rate is denoted by θ_1, θ_2 and θ_3, respectively.

this link. This allocation is certainly feasible and gives even share to each user, and it is also maximal in this regard. On the other hand, this allocation would waste resources (so it would go against our objective for "efficiency" set out above): observe that user $(3, 5)$ could attain higher rate (a rate of 1 unit) if he/she would be able to use the full capacity of link $(3, 5)$, but this would violate our *fairness principle* that we assign even share to each user.

Max-min fair resource allocations. It seems that we need to come up with a better notion for defining the way to balance between the, seemingly contradictory, requirements of feasibility, efficiency, and fairness [115]. Taking ideas from axiomatic theories of fairness [131], economy (Atkinson's index [9, 213]), game theory (Nash bargaining [153], Shapley value [188]), sociology [131], political philosophy [111], and multi-user computer systems [108, 110, 131], fair resource allocations are usually associated with the below four *fairness principles* (see a comprehensive discussion in [110, Appendix D] and a unifying treatment in [131]):

- *Sharing incentive:* a fair allocation incentivizes users to share resources, by ensuring that no user is better off in a system in which resources are statically and equally partitioned.
- *Strategy-proofness:* users cannot improve their allocation by lying about their specific requirements.
- *Envy-freeness:* no user would want to trade his/her allocation with that of another user.
- *Efficiency:* it is not possible to improve the allocation of a user without decreasing the allocation of some other user (*Pareto-efficiency*).

It has been shown that in the context of bandwidth allocation in capacitated networks these fairness principles boil down to the *max-min fair* allocation strategy [107] (but see also [131]), defined loosely speaking as an allocation whereby "there is no way to make any person better off without hurting anybody else who is already poorer" (see later for a formal definition).

It is by far not evident whether such a max-min fair allocation exists in a specific

context, like the fair bandwidth allocation problem [133]. Still, for the case when the paths for users are fixed, existence of a max-min fair rate allocation is indeed guaranteed [107].

Example 2.2. Assume that in the sample network of Example 2.1, path $1 \to 4 \to 5$ is assigned to user $(1, 5)$, path $2 \to 4 \to 5$ to user $(2, 5)$, and the direct path $3 \to 5$ to user $(3, 5)$, respectively. Then, the max-min fair allocation is given by the vector $[1/2, 1/2, 1]$, where coordinates specify the bandwidth share received by each user, as per the ordering of users in Fig. 2.2. Observe that this allocation indeed possesses the max-min fair property in that no users could get better rate without taking away bandwidth from some other user whose share is already smaller or equal. For instance, if we were to increase the rate of user $(1, 3)$ from $1/2$ to, say, $3/4$, then we would need to decrease the share of user $(2, 3)$ from $1/2$ to $1/4$ as the aggregate capacity that is made available to the users is constrained at the bottleneck edge $(4, 5)$ at 1 unit.

The fixed-path model. Curiously, the actual selection of paths influences the emergent max-min fair allocation, and bottlenecks, to a great extent.

Example 2.3. Consider Example 2.1. If the path of user $(3, 5)$ is changed to $3 \to 4 \to 5$, then the max-min fair rate vector turns to $[1/3, 1/3, 1/3]$ (another "even share" bandwidth allocation). It is possible to extend this "single-path" formulation to a (limited) "multi-path" setting under the assumption that the traffic splitting ratio at paths' branching nodes is known in advance: if we assign both paths $3 \to 5$ and $3 \to 4 \to 5$ to user $(3, 5)$ with the restriction that traffic must be split equally between the two paths, then the max-min fair allocation ends up being $[2/5, 2/5, 2/5]$ and the (shared) bottleneck is again link $(4, 5)$ [26].

Max-min fair allocations in this fixed-path model are strongly dependent on the specific routes assigned to users and this goes against the very fairness principles enumerated above; even though users may not envy each other's *rate* but they may certainly envy each other's *routes* and may rightfully ask for a routing that would favor their desire for more bandwidth. Consequently, the *"fixed-path" version of the fair bandwidth allocation problem violates basically all of the principles for fairness*, in that

- it may not provide incentive for sharing as users will strive for a routing that maximizes their max-min share (e.g., user $(3, 5)$ will want exclusive access to both paths $3 \to 4 \to 5$ and $3 \to 5$ as his/her max-min share becomes 2 units in this case);
- it is not strategy proof as users will generally lie about their bandwidth requirements to get a routing that maximizes their share;
- it is not envy-free, since a user may not be satisfied with the routing, and the resultant max-min share, he/she receives;
- and finally it is not (Pareto-)efficient as it is dependent on the exact paths and traffic splitting ratios assigned to the users and there may exist other assignments that would lead to better utility.

The main objective of this Chapter is to sidestep this adverse dependence of bandwidth allocations on particular assignment of paths; namely, the *general formulation for the max-min fair bandwidth allocation problem* asks for an allocation that is independent from particular routings. The idea is that we would find a max-min fair allocation that is only dependent on the specifics of the network (topology and link capacities) and not on some random fixed paths and, consequently, would fulfill all the fairness principles, i.e., sharing incentive, strategy-proofness, envy-freeness, and Pareto-efficiency.

2.2.1 Model and Notation

In the mathematical model, we are given a *network configuration* comprising *(i)* a *directed graph* $G(V, E)$ with nodes V and edges, links, or directed arcs[2], E, where $n = |V|$ and $m = |E|$; *(ii)* the column m-vector of (finite) link *capacities* $c = [c_{ij} > 0 : (i, j) \in E]$; *(iii)* a set of *users*, represented by a set of unique source-destination pairs $(s_k, d_k) : k \in \mathcal{K}$; and *(iv)* a set of paths[3] $\mathcal{P}_k$ available to each user $k \in \mathcal{K}$. In our model, $\mathcal{P}_k$ may contain only a single path for each user, in which case the model simplifies into the conventional fixed single-path model, a set of "good" paths assigned by the operator (e.g., using the k-shortest path algorithm), or, at the extreme setting, it may contain *all* directed paths from $s_k \to d_k$ in G_c; the splitting ratios do not need to be fixed in any of these cases. The theory we present will apply to each of these settings. Let p_k denote the number of paths for k and let p be the number of *all* paths. Finally, let P_k describe path-arc incidence matrix corresponding to $\mathcal{P}_k$; here, P_k has a p_k columns, one for each directed path in G that can be utilized by user k for sending traffic, and m rows, one for each arc $(i, j) \in E$. Then, the entry in P_k corresponding to path P and edge (i, j) equals 1 if P traverses (i, j) in the same direction as (i, j) is oriented and zero otherwise.

We shall use the short-hand notation G_c to mean a particular network configuration, with the graph, capacities, and users included. The below mild regularity condition then gives a useful characterization of the network configurations that admit a reasonable definition for the fair bandwidth allocation problem.

Definition 2.1. A network configuration G_c is *regular*, if

- a path exists in G_c from s_k to d_k for each $k \in \mathcal{K}$ and
- all edge capacities are finite and strictly positive.

It is easy to see that any network configuration can be reduced to a collection of regular network configurations by rewriting infinite link capacities with a capacity "large enough", removing edges with zero capacity, and clearing disconnected users.

Feasible routings. Next, we define formally the set of feasible routings, i.e., assignments of paths to users together with respective sending rates, supported by a particular network configuration. Let u_P denote the amount of traffic, or *flow*, sent by user k to path $P \in \mathcal{P}_k$ and let u_k denote the column-vector whose components are $u_P : P \in \mathcal{P}_k$:

$$u = [u_k : k \in \mathcal{K}] \in \mathbb{R}^{p_1} \times \mathbb{R}^{p_2} \times \ldots \times \mathbb{R}^{p_K} = \mathbb{R}^p \ .$$

The Euclidean space $\mathbb{R}^p$ will be called the *flow space*. See a summary of notations in Table 2.1.

With this notation in mind the below definition gives the set of all feasible routings supported by some network configuration.

Definition 2.2. Given a network configuration, the *flow polytope* $M(G_c) \subset \mathbb{R}^p$ is the set of admissible path-flows, subject to link capacities and non-negativity constraints:

$$M(G_c) = \{u : \sum_{k \in \mathcal{K}} P_k u_k \leq c, \ u \geq 0\} \ . \tag{2.1}$$

[2]In the rest of this Chapter we use the terms "link" and "arc" interchangeably.

[3]Path-flow formulation is chosen only for convenience. The results apply equally to the arc-flow formulation.

Convex polyhedra. Readers proficient in network flow theory might find the formulation (2.1) familiar, since $M(G_c)$ is in fact the set of feasible solutions for the family of *multicommodity flow problems* [186]. Here, the constraint $\sum_k P_k u_k \leq c$ requires that flows do not over-utilize link capacities and $u \geq 0$ disallows negative flows. Note that the term "polytope" implies a geometric concept, in particular, a polytope is a compact convex set whose faces are "flat" (hyper)planes, a higher-dimension geometric generalization of the well-known concepts of polygons (in 2D) or (Platonian) solids (in 3D). Following are some important definitions form convex analysis that we shall rely on below [234] [87] [19] [186].

Definition 2.3. Let $\mathbb{R}^n$ be the n-dimensional Euclidean space.

- *Polyhedra:* A set $P \subseteq \mathbb{R}^n$ is a *polyhedron* if it arises as the intersection of finitely many closed half-spaces: $P = \{x : Ax \leq b\}$ for some $m \times n$ matrix A and column m-vector b (*half-space representation*). An inequality $ax \leq b$ for P is a *valid inequality* if $\forall x \in P : ax \leq b$ holds.
- *Properties of polyhedra:* A polyhedron P is a *convex* set: for any $x_1, x_2 \in P$: $\lambda x_1 + (1-\lambda)x_2 \in P$. In addition, P is *down-monotone* if for any $x \in P$ and for any $0 \leq y \leq x : y \in P$. P is *bounded* if it does not contain a ray $\{x + \lambda y : \lambda \geq 0\}$. A *compact* (closed and bounded) polyhedron is called a *polytope*.
- *Extreme points:* The *convex combination* $\mathrm{Conv}\{x_1, \ldots, x_s\}$ of points $\{x_1, x_2, \ldots, x_s\}$ in $\mathbb{R}^d$ is defined as

$$\mathrm{Conv}\{x_1, \ldots, x_s\} = \left\{ x : \exists \lambda_1, \ldots, \lambda_s, \lambda_i \geq 0, \text{ where } x = \sum_{i=1}^{s} \lambda_i x_i \text{ and } \sum_{i=1}^{s} \lambda_i = 1 \right\} .$$

  Given a polytope P, some $x \in P$ is an *extreme point* of P if it cannot be generated as the convex combination of two distinct points in P. Any polytope $P = \{x : Ax \leq b\}$ is equivalently described by the convex-combination of its extreme points $x_1, \ldots, x_s$: $P = \mathrm{Conv}\{x_1, \ldots, x_s\}$ (*vertex-representation*).
- *Operations on polytopes:* An *affine projection* of a polytope P through an *affine map* $\pi(x) = Ax + b$ is a set $\pi(P) = \{\pi(x) : x \in P\}$; an affine projection of a polyhedron is again a polyhedron and if P is bounded then $\pi(P)$ is also bounded. The *scalar multiple* of a polytope $P = \{x : Ax \leq b\}$ is defined as $\lambda P = \{x : Ax \leq \lambda b\}$.
- *Triangulations:* The *boundary* ∂P of P consists of the set of points $x \in P$ for which one or more inequalities in $Ax \leq b$ hold with equality. A *simplex* is a d-dimensional polytope arising as the convex combination of exactly $d+1$ *affinely independent* extreme points. A *polyhedral partition* of P is a set of disjunct (apart from the boundaries) polytopes $Q_i : i \in \{1, \ldots, q\}$ so that $P = \bigcup_i Q_i$. A *triangulation* is a polyhedral partition $Q_i : i \in \{1, \ldots, q\}$ so that each Q_i is a simplex. A *boundary-triangulation* is a triangulation where the extreme points of Q_i do not introduce interior points, i.e., each Q_i is a convex combination of some subset of the extreme points of P.

Easily, $M(G_c)$ is an intersection of finitely many half-spaces by (2.1) and consequently it is indeed a polyhedron. Additionally, it is also bounded and full-dimensional if G_c is regular.

Feasible and fair rate allocations. Given a routing $u_k, k \in \mathcal{K}$, the (total) *rate*, or *throughput*, of user k equals the sum of the flows sent by user k to the paths available to

Table 2.1: Notation.

$G(V, E)$	a directed graph, with the set of nodes V ($\|V\| = n$) and the set of directed arcs E ($\|E\| = m$)
c	the column m-vector of arc capacities
(s_k, d_k)	source-destination pairs, or users, for $k \in \mathcal{K} = \{1, \ldots, K\}$
$\mathcal{P}_k$	the set of $s_k \to d_k$ paths assigned to some user $k \in \mathcal{K}$
P_k	an $m \times p_k$ path-arc incidence matrix for the paths $\mathcal{P}_k$ of user k
P_k^{ij}	the row of P_k corresponding to arc $(i, j) \in E$
p_k	the number of paths for user k, $p_k = \|\mathcal{P}_k\|$
p	number of all paths, $p = \sum_{k \in \mathcal{K}} p_k$
u_P	path-flow routed over path $P \in \mathcal{P}_k$
u_k	a column-vector, whose components are $u_P : P \in \mathcal{P}_k$ for some $k \in \mathcal{K}$ (whether we mean u_k or u_P will always be clear from the context)
u	a routing, a column p-vector $u = [u_k : k \in \mathcal{K}]$
θ	a traffic matrix, a column K-vector $\theta = [\theta_k : k \in \mathcal{K}]$
θ_k	the amount of traffic requested by user $k \in \mathcal{K}$
$M(G_c)$ or M	flow polytope, the set of path flows on $\mathcal{P}$ subject to non-negativity and capacity constraints
$T(G_c)$ or T	demand or throughput polytope, the set of flow rates realizable in G_c over $\mathcal{P}$ subject to capacity constraints
1^T	a vector of all 1s of proper size
$\mathcal{T}$	throughput mapping $\mathcal{T}$, a $\mathbb{R}^p \mapsto T$ function $\mathcal{T}(u) = [\theta_k = 1^T u_k : k \in \mathcal{K}]$
$\mathcal{C}$	a cut, a set of edges $\mathcal{C} \subseteq E$ whose removal from the network would disconnect all directed s_k to d_k paths for at least one user $k \in \mathcal{K}$
$\mathcal{K}_\mathcal{C}$	the set of users disconnected by some cut $\mathcal{C}$

k: $\theta_k = \sum_{P \in \mathcal{P}_k} u_P = 1^T u_k$. We collect users' rates into a column K-vector $\theta = [\theta_k : k \in \mathcal{K}]$ for ease of notation; the vector $\theta \in \mathbb{R}^K$ is called a *traffic matrix* and $\mathbb{R}^K$ is called the *throughput space*. Given a traffic matrix $\theta = [\theta_k : k \in \mathcal{K}]$, we say that a routing u *realizes* θ if $u \in M(G_c)$ and $\theta_k = 1^T u_k$ for each $k \in \mathcal{K}$. Since the mapping from the flow space to the throughput space will often come up in the context of this and the subsequent Chapter, we introduce a distinct notation and terminology below.

Definition 2.4. The *throughput mapping* $\mathcal{T}$ is a $\mathbb{R}^p \mapsto \mathbb{R}^K$ function $\mathcal{T}(u) = Qu$, where Q is a $K \times p$ matrix, the elements in kth row of Q are all 1 at positions $\sum_{l<k} p_l + 1, \ldots, \sum_{l \le k} p_l$ and all zero otherwise.

Using this notation, we can give a series of increasingly stronger definitions for fair rate allocations:

Definition 2.5. Given a network configuration G_c, a rate allocation $\theta = [\theta_k : k \in \mathcal{K}]$ is

- *feasible*, if there exists a routing u that realizes θ: $u \in M(G_c)$ and $\theta_k = 1^T u_k, k \in \mathcal{K}$;
- *non-dominated for some user* k, if changing the allocation from θ_k to $\theta_k + \epsilon$ for user k while fixing the rate of other users would be infeasible for any $\epsilon > 0$;
- *(strictly) Pareto-efficient*, if all the users are non-dominated at θ; and finally

Algorithm 2.1 Generalized water-filling algorithm for network configuration G_c.

1: $\mathcal{B} = \emptyset$; $\theta = 0$
2: **while** $\mathcal{B} \neq \mathcal{K}$
3: $\quad \lambda \leftarrow \max_{\lambda>0}\{\lambda : \theta_k + \lambda \text{ is feasible for each } k \in \mathcal{K} \setminus \mathcal{B}\}$
4: $\quad \theta_k \leftarrow \theta_k + \lambda$ for each $k \in \mathcal{K} \setminus \mathcal{B}$
5: $\quad \mathcal{B} \leftarrow \{k \in \mathcal{K} : k \text{ is non-dominated at } \theta\}$
6: **end while**

- *max-min fair*, if it is Pareto-efficient and every other feasible allocation has the property that the rate of a user can be increased only at the price of decreasing the rate of some other user that already receives a smaller, or equal rate:

$$\text{for each feasible rate allocation } \theta' : \theta'_k > \theta_k \Rightarrow \\ \exists l \in \mathcal{K} \setminus \{k\} \text{ so that } \theta'_l < \theta_l \text{ and } \theta_l \leq \theta_k \ . \quad (2.2)$$

In this setting, the efficiency principle is embodied by the requirements for non-dominatedness and Pareto-efficiency while envy-freeness is guaranteed by the max-min principle.

The water-filling algorithm in the fixed-path model. A way to obtain the max-min fair allocation itself in the fixed-path model is the *water-filling algorithm*, an iterative rate augmentation procedure to obtain a bandwidth allocation that admits a so called *bottleneck argumentation* and therefore is guaranteed to be max-min fair [26]. In each iteration of the water-filling algorithm users' rates are increased at the same pace until some edge gets saturated, at which point we fix the rate of the users whose path traverses the saturated edge and assign the edge as a *bottleneck* for these users, and then keep on increasing the rate of unblocked users until eventually all users get blocked. In the conventional setting of the fixed-path model the correct termination of the algorithm is trivially guaranteed by that each user has a single path (or multiple paths with fixed splitting ratios) [26]; observe, however, that such guarantees do not so trivially exist when paths become problem variables (cf., e.g., the search in line 3).

Algorithm 2.1 gives the formal description of the water-filling algorithm (see later for why we call it the "Generalized water-filling algorithm" already at this point).

The water-filling algorithm is guaranteed to terminate in a max-min fair allocation θ in $O(|\mathcal{K}|)$ steps; the argumentation goes on by showing that the bottleneck edge e_k assigned by the algorithm to each user $k \in \mathcal{K}$ has the property that *(i)* e_k is filled to capacity at θ and *(ii)* the user k has the maximum rate amongst the users whose path traverses e_k; these properties together result in a rate allocation that is Pareto-efficient and fulfills (2.2), i.e., is max-min fair [26].

2.2.2 Problem Formulation

The max-min fair bandwidth allocation problem is concerned with finding an allocation of rates to users that fulfills all four fairness principles set out above: sharing incentive, strategy-proofness, envy-freeness, and Pareto-efficiency. The fixed-path version, where the set of paths P_k available to each user k is limited to a single $s_k \to d_k$ path, is adequately handled in all undergraduate text books on networking [26]; however, we have seen that this strategy might not comply with the fairness principles in that the dependence on

a particular set of, from the users' perspective, arbitrary, routes may still induce envy between users.

The *general formulation for the max-min fair bandwidth allocation problem* asks for a fair resource allocation that would be independent of any routings whatsoever [169–171, 173]. This problem was first raised in [133, Section "When bottleneck and water-filling become less obvious"], in the hope that it would fix the fairness issues of the fixed-path model. Then, the question whether a bottleneck argumentation exists that would intuitively generalize the analogous notions from the fixed-path model to the routing-independent generic model was left open. This formulation is the main concern in this Chapter; for the purposes of the subsequent discussion we give the formal problem statement as follows.

Definition 2.6. Given a network configuration G_c, the general max-min fair bandwidth allocation problem is concerned with finding an allocation of rates θ that is

- feasible: $\exists u \in M(G_c)$ so that $\theta_k = 1^T u_k$ for each $k \in \mathcal{K}$;
- Pareto-efficient: increasingthe rate of any user $k \in \mathcal{K}$ from θ_k to $\theta_k + \epsilon$ while fixing the rate of the rest of the users would be infeasible for any $\epsilon > 0$; and
- envy-free: no user could get larger rate without decreasing the rate of some other user that is already smaller or equal, see Eq. (2.2).

This definition is now independent of any particular selection of routings, as the only input is the network configuration (i.e., the graph, link capacities, and users) itself. Correspondingly, the water-filling algorithm, and the related constructive schemes to prove the existence of a max-min fair allocation, cannot be extended to the general case naively, since these constructions depend on a particular routing (recall Algorithm 2.1).

2.3 General Max-min Fair Bandwidth Allocation

Below, we tackle the *general formulation for the max-min fair bandwidth allocation problem* by identifying a bandwidth allocation scheme that is dependent only on the specifics of the network configuration without having to fix the paths of the users beforehand in any ways [169–171, 173]. First, we restate an earlier finding from [133] that such an allocation is guaranteed to exist in any network, but this time adopting an unconventional, purely geometric approach. Our new approach will then allow us to go beyond the insights that could be attained by lexicographic optimization in [133]; in particular, we give a bottleneck argumentation for the general setting and show how intuitively it generalizes the concept of bottlenecks from the fixed-path model.

2.3.1 The Feasible Set of the Bandwidth Allocation Problem

Our strategy to solve the general formulation for the max-min fair bandwidth allocation problem is to characterize the feasible set of the problem and, provided that the feasible set is compact and convex, use the result from [133, Theorem 1] to show that the max-min fair allocation exists and it is unique.

The bandwidth allocation problem asks for a set of rates, one particular scalar rate assigned to each user, that fulfills the criteria of feasibility, Pareto-efficiency, and envy-freeness (cf. Definition 2.6). Here, the latter two requirements, Pareto-efficiency and envy-freeness, merely point to certain allocations that are somehow desirable from an operational standpoint, and as such can be seen as "objectives" to maximize over some

set of feasible rates, but the set of admissible allocations is purely dictated by the first requirement, feasibility. This in turn is, recall, informally stated as follows: a rate allocation is feasible if there is a routing that realizes it, subject to capacity constraints, as the sum of path-flows for each user; here, the feasible path-flows themselves are characterized by the flow polytope as per Definition 2.2.

With this observation in mind, we can describe the set of *feasible rate allocations*, which we shall denote for a particular network configuration G_c as $T(G_c)$, as follows:

Definition 2.7. Given a graph configuration G_c. the set of feasible rate allocations in G_c is defined as follows:

$$T(G_c) = \{\theta : \exists u \in M(G_c) \text{ so that } \theta_k = 1^T u_k \text{ for each } k \in \mathcal{K}\} \subset \mathbb{R}^K \ . \tag{2.3}$$

In this setting, the set $T(G_c)$ is generally restricted only on the input graph configuration G_c, but it is completely independent of particular routings. Our critical observation is that $T(G_c)$ arises as an affine projection of the flow polytope $M(G_c)$ and as such, it is itself a convex polytope. The formal result is as follows [29, 171, 177]:

Theorem 2.8. For any network configuration G_c, the feasible set $T(G_c)$ of the rate allocation problem is a convex polyhedron. Additionally, if G_c is regular then $T(G_c)$ is bounded, full-dimensional, and down-monotone. In general the size of the half-space representation of $T(G_c)$ may grow exponentially with the network size (irrepresentability).

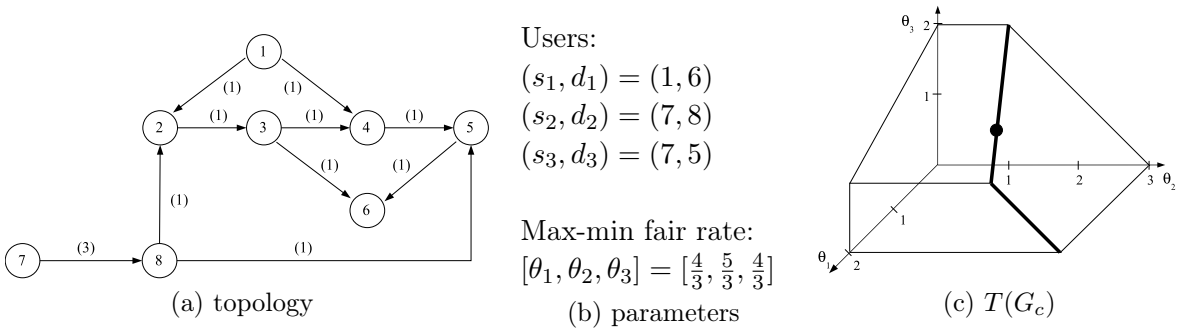
Proof. From (2.3) it follows that $T(G_c)$ is the affine projection of $M(G_c)$ through the affine map $\pi(u) = \Pi u = [\pi_k(u_k) : k \in \mathcal{K}]$, $\pi_k(u_k) = 1^T u_k$. As such, it is a polyhedron by Definition 2.3 and, provided that G_c is regular, it is also bounded and full-dimensional by that $M(G_c)$ is also bounded and full-dimensional and the projection matrix Π is of full row rank. Finally, it is also fairly easy to see that $T(G_c)$ is down-monotone (this is also called the "free-disposal property" in [133]): if some allocation of rates θ is feasible then any allocation $0 \leq \tau \leq \theta$ it dominates is also feasible. Finally, regarding irrepresentability: in [29] we show a network configuration in which both the half-space and the vertex representation of T grows as $\Omega(2^K)$ with the number of users K. Accordingly, in general no polynomial size description for $T(G_c)$ exists. □

We shall call $T(G_c)$ the *throughput polytope* henceforth and we shall usually assume the regularity of G_c.

Example 2.4. The throughput polytope for the sample network in Example 2.1 is given in Fig. 2.2c and is formally specified as follows:

$$T(G_c) = \{[\theta_1, \theta_2, \theta_3] : \theta_1 + \theta_2 + \theta_3 \leq 2 \tag{2.4}$$
$$\theta_1 + \theta_2 \leq 1 \tag{2.5}$$
$$\theta_1, \theta_2, \theta_3 \geq 0\} \tag{2.6}$$

For instance, the constraint $\theta_1 + \theta_2 \leq 1$ confines the aggregate rate of user $(1, 5)$ and $(2, 5)$ at 1 unit; this constraint comes from the fact that these users share the link $(4, 5)$ and the capacity of that link, 1 unit, does not allow them to get higher aggregate rate. Similarly, $\theta_1 + \theta_2 + \theta_3 \leq 2$ comes from that the total traffic needs to be routed through the cut formed by the links $\{(3, 5), (4, 5)\}$ and the aggregate capacity of this cut, 2 units, presents an impenetrable bottleneck in rate allocation.



(a) topology (b) parameters (c) $T(G_c)$

Figure 2.3: Another sample network and the associated set of feasible rates. Edge capacities are marked in parentheses. Pareto-efficient allocations are marked by bold lines, while the max-min fair point is denoted by a black dot.

The observation that certain constraints, or valid inequalities, and certain cuts are fundamentally associated with critical resource shortages in the network will be fundamental in the below in developing our bottleneck argumentation for the general bandwidth allocation problem. In the rest of this Chapter, we shall use the below, slightly more complex example to demonstrate the importance of this observation.

Example 2.5. Consider the network configuration given in Fig. 2.3. The corresponding throughput polytope is as follows:

$$T(G_c) = \{[\theta_1, \theta_2, \theta_3] : \theta_1 \leq 2 \tag{2.7}$$
$$\theta_2 + \theta_3 \leq 3 \tag{2.8}$$
$$\theta_1 + 2\theta_3 \leq 4 \tag{2.9}$$
$$\theta_1, \theta_2, \theta_3 \geq 0 \ \} \tag{2.10}$$

In the below discussions we shall usually consider the below half-space representation of $T(G_c)$:

$$T(G_c) = \{\theta \geq 0 : \beta_i^T \theta \leq b_i, i \in \mathcal{I}\} \ , \tag{2.11}$$

where $\mathcal{I}$ is a (finite) index set and for each $i \in \mathcal{I}$ it holds that $\beta_i^T \geq 0$ and b_i is a positive scalar. Such a half-space representation is guaranteed to exist by Theorem 2.8; in particular, $\beta_i^T \geq 0$ and $b_i > 0$ are guaranteed by down-monotonity.

The remarkable observation here is that the constraint matrix is non-negative, $\beta_i^T \geq 0$, and the right-hand-side is strictly positive, $b > 0$. There exists a far-reaching characterization of the throughput polytope that explains why this is the case, in that it can be shown that *each* valid inequality of $T(G_c)$ arises as shortest-path lengths over some non-negative weights assigned to the edges in G_c; this observation is sometimes referred to as the "Japanese Theorem" on the traces of [105] and [162], see also [186].

Proposition 2.9. Let G_c be a regular network configuration and let $T(G_c)$ be the corresponding throughput polytope. Then, an inequality $\beta^T \theta \leq b$ is valid for $T(G_c)$ if and only if there exist non-negative weights $w^T = [w_{ij} : (i, j) \in E]$ on the edges of $G(V, E)$ so that $\beta^T = [\beta_k : k \in \mathcal{K}]$ is less than, or equal to the length of the shortest path from s_k to d_k over the edge weights w for each user $k \in \mathcal{K}$ and $b = w^T c$.

Proof. Since $T(G_c)$ is the affine projection of $M(G_c)$ through the affine map $\pi(u) = [1^T u_k : k \in \mathcal{K}]$, we can apply Černikov's block-elimination method [234] to $M(G_c)$ to obtain that row K-vectors β^T and row m-vectors w^T lying in the projection cone

$$W(G_c) = \{[\beta^T, w^T] : \sum_{(i,j)\in P} w_{ij} \geq \beta_k \qquad \forall k \in \mathcal{K}, \forall P \in \mathcal{P}_k \tag{2.12}$$

$$w^T \geq 0 \ \} \tag{2.13}$$

generate *all* the valid inequalities of $T(G_c)$ as follows:

$$[\beta^T, w^T] \in W(G_c) \Leftrightarrow \forall \theta \in T(G_c) : \beta^T \theta \leq w^T c \ .$$

In fact, it is enough to take the inequalities generated by the *extreme rays* of $W(G_c)$. Thus, the representation (2.11) contains only finitely many half-spaces [234]. Observe that here vectors w can be thought of as non-negative weights and β_ks as upper bounds on the weight of the shortest path from s_k to d_k over the edge weights w for each $k \in \mathcal{K}$, which concludes the proof. □

Example 2.6. Recall the sample network in Example 2.5 and consider the valid inequality $\theta_1 + 2\theta_3 \leq 4$ in the half-space representation (2.7)–(2.10) of the throughput polytope. It is easy to see that this inequality is generated by the weight assignment $w_{2,3} = w_{4,5} = 1$, $w_{8,5} = 2$, and all zero otherwise, and the resultant shortest path weights are $\beta_1 = 1$ for user $(1, 6)$, $\beta_2 = 0$ for user $(7, 8)$, and $\beta_3 = 2$ for user $(7, 5)$. The reader easily checks that the rest of the inequalities have their own generating weights too; note that one can find such a generating weight set for any valid inequality by solving a linear program over the projection cone (2.12)–(2.13) (see later in Section 2.4.2).

2.3.2 Max-min Fair Allocation on the Throughput Polytope

Next, we establish the existence of a well-defined solution for the general max-min fair bandwidth allocation problem. In particular, we use the below result from [133, Proposition 1 and Theorem 1]:

Proposition 2.10. If a set X is convex and compact, then there exists a max-min fair allocation on X and it is unique.

In the previous section, we have shown that the set of feasible rate allocations over a regular network configuration is a polytope, which is by regularity convex and compact (see Theorem 2.8). This gives rise to the below result.

Corollary 2.11. Given a regular network configuration, a solution to the general max-min fair bandwidth allocation problem exists and it is unique.

2.4 Generalized Bottlenecks

We now turn to discuss the way bottlenecks arise in the context of the general max-min fair bandwidth allocation problem. Recall, a bottleneck argumentation is crucial in the context of networking as bottlenecks point at certain critical shortages of resources in a network that adversely constrain users' achievable rates, and because they also substantiate a fast iterative algorithm, the water-filling algorithm, to find the max-min fair allocation itself. Note that the max-min fair allocation could still be found, by max-min programming, even in the absence of a bottleneck argumentation, but water-filling is much faster and more intuitive [133].

2.4.1 Geometric Interpretation

First, we build a geometric understanding by developing adequate bottleneck argumentations for increasingly complex fairness notions, starting from the set of feasible rate allocations and iterating from simpler notions, like non-dominatedness and Pareto-efficiency, to fully-fledged max-min fairness (cf. Definition 2.5). Our key observation is that valid inequalities of the throughout polytope provide a purely geometric framework to generalize bottlenecks from the fixed-path model to the general one and the defining properties of bottlenecks readily find their geometric counterparts in this framework.

Above, we have developed the insight that different fairness notions merely state at certain subsets of the feasible rate allocation set, that is, the throughput polytope, and as such can be approached as simple objective functions that embody the particular notion of fairness under consideration, which is to be maximized over the feasible set to obtain the fair rate allocation itself. This insight then drives us to characterize these "fair subsets" of the feasible rate allocation space in terms of certain *touching hyperplanes*, or valid inequalities, of the throughput polytope, on the basis that such touching hyperplanes in convex analysis provide the mathematical framework to identify the optimal feasible solutions of linear and convex programs [19].

Non-dominated rate allocations. Recall, the rate vector θ is non-dominated for some user k if there is no allocation with strictly larger share for k when leaving the share of the rest of the users intact. The following result gives a bottleneck argumentation in the geometric sense for such non-dominated allocations.

Theorem 2.12. Consider a feasible rate allocation $\theta \in T(G_c)$ and let $\mathcal{N} \subseteq \mathcal{K}$ denote the set of non-dominated users at θ. Then, $\mathcal{N} \neq \emptyset$ if and only if there exists an inequality $\beta^T\theta \leq b$ that is

- *valid:* for each $\theta' \in T(G_c) : \beta^T\theta' \leq b$,
- *tight:* $\beta^T\theta = b$, and
- *complementary:* $(\beta)_k > 0$ if and only if $k \in \mathcal{N}$.

Proof. Of course, if no user is dominated at θ then there can be no valid inequality that were tight at θ, i.e, $\nexists \beta^T, b : \beta^T\theta' \leq b$ for all $\theta' \in T(G_c)$ but $\beta^T\theta = b$. To prove the other direction, let $T(G_c) = \{\theta \geq 0 : \beta_i^T\theta \leq b_i,\ i \in \mathcal{I}\}$ and assume that $\mathcal{N} \neq \emptyset$. Furthermore, let $\mathcal{B}$ be the index set of constraints binding at θ: $\mathcal{B} = \{i \in \mathcal{I} : \beta_i\theta = b_i\}$, and let $\beta^T = \sum_{i\in\mathcal{B}} \beta_i^T$ and $b = \sum_{i\in\mathcal{B}} b_i$. Note that $\mathcal{N} \neq \emptyset \Leftrightarrow \mathcal{B} \neq \emptyset$.

First, $\beta^T\theta' \leq b$ is valid for $T(G_c)$ since it is the non-negative sum of valid inequalities for $T(G_c)$. This proves the first claim. Second, $\beta^T\theta' \leq b$ is tight at θ, $\beta^T\theta = b$, since it is the sum of valid inequalities binding at θ, which proves the second claim. To prove complementarity, using $\beta_i \geq 0$ for each $i \in \mathcal{I}$ we write: $(\beta)_k = 0$ for $k \in \mathcal{K} \Leftrightarrow (\beta_i)_k = 0$ for all constraints binding at $\theta \Leftrightarrow \exists \epsilon > 0$ and small enough so that $\theta + \epsilon e_k \in T(G_c) \Leftrightarrow$ user k is dominated at θ. This concludes the proof. □

Example 2.7. Consider the rate allocation $\theta = [2, 0, 1]$ in the sample network of Example 2.5, where user $(s_1, d_1) = (1, 6)$ receives 2 units of bandwidth and user $(s_3, d_3) = (7, 5)$ receives 1 unit, and user $(s_2, d_2) = (7, 8)$ does not get any capacity at all. This allocation is clearly feasible, as user $(1, 6)$ may use the paths $1 \to 2 \to 3 \to 6$ and $1 \to 4 \to 5 \to 6$ by splitting its traffic equally and user $(7, 5)$ may use $7 \to 8 \to 5$. Furthermore, users $(1, 6)$ and $(7, 5)$ are non-dominated at this rate allocation as they cannot voluntarily increase

their share without taking away capacity from each other, while user $(7, 8)$ is dominated as there are 2 units of free capacity available along the direct path $7 \to 8$. To obtain the corresponding valid inequality, we sum up the two binding constraints (2.7) and (2.9) to obtain $\theta_1 + \theta_3 \leq 3$. Observe how this inequality is valid and tight at θ, and also that the coefficients β_k are strictly positive precisely for the non-dominated users $\mathcal{N} = \{1, 3\}$.

The above complementarity property will motivate us to call such valid inequalities as "geometric bottlenecks". Indeed, in the above example the inequality $\theta_1 + \theta_3 \leq 3$ is exactly the upper bound on our capacity to increase the aggregate rate of user 1 and 3 any further from $\theta = [2, 0, 1]$, and as such, marks a critical shortage of resources in the network for these two users. Should they ask for more bandwidth, the operator could always point user 1 and 3 at this inequality and argue that it is impossible to augment the bandwidth allocation of any one of them without hurting the other one.

Pareto-efficient rate allocations. A Pareto-efficient rate allocation is such that any user is either at its (single-commodity) maximum flow, so the current rate cannot be increased at all, or otherwise increasing the rate is possible only at the expense of decreasing the rate of some other user. It is then easy to dissect Pareto-efficiency to non-dominatedness.

Observation 2.13. A rate allocation $\theta \in T(G_c)$ is (strictly) Pareto-efficient if and only if all users are non-dominated at θ.

The following result, which pinpoints the "bottleneck inequalities" for Pareto-efficient allocations, is easily seen to be a simple application of Theorem 2.12 and Observation 2.13.

Theorem 2.14. A rate allocation $\theta \in T(G_c)$ is Pareto-efficient, if and only if there exists an inequality $\beta^T\theta \leq b$ that is

- *valid:* for each $\theta' \in T(G_c) : \beta^T\theta' \leq b$,
- *tight:* $\beta^T\theta = b$, and
- *strictly positive:* $\forall k \in \mathcal{K} : (\beta)_k > 0$.

This suggests a simple (water-filling-like) algorithm to search for a Pareto-efficient allocation in $T(G_c)$: in each iteration increase the rate of dominated users at the same pace as long as it is possible; eventually all the users will be blocked and so a valid inequality $\beta^T\theta \leq b$ with all strictly positive $(\beta)_k$ coordinates must be binding at the resultant point; hence the emergent θ is Pareto-efficient.

Example 2.8. In the sample network of Example 2.5, the two line segments joining the points $[2, 3, 0]$ and $[2, 2, 1]$, respectively $[2, 2, 1]$ and $[0, 1, 2]$, contain all the Pareto-efficient points (see the bold line segments in Fig. 2.3c). The reader will easily generate the valid inequalities for these points as per Theorem 2.14.

Max-min fair rate allocation. From the aspect of fairness, Pareto-efficiency is a somewhat weak, although clearly desirable criterion; desirable because it avoids the wastage of resources non-dominatedness generally allows for, and weak because Pareto-efficiency permits allocations where one user gets everything (for instance the allocation $[0, 0, 2]$ is Pareto-efficient in the sample network of Example 2.1). The concept of max-min fairness is based on the idea to pick the "fairest" Pareto-efficient allocation, where fairness itself is manifested by the premise that "there is no way to make any person better off without

hurting anybody else who is already poorer" (see Definition 2.6). Correspondingly, a max-min fair allocation, if it exists, provably provides sharing incentive, it is strategy-proof, and envy-free. Existence of a solution in turn for the context of the general max-min fair bandwidth allocation problem is guaranteed by Corollary 2.11.

What remained to be done is to find an adequate notion for bottleneck inequalities for max-min fairness, on the traces of Theorem 2.12 and Theorem 2.14. It turns out, however, that this time our characterization involves not just one but exactly K valid inequalities, one for each user. Consider the below claim.

Theorem 2.15. A rate allocation $\theta \in T(G_c)$ is max-min fair, if and only if for each user $k \in \mathcal{K}$ there exists an inequality $\beta^T\theta' \leq b$ that is

- *valid:* for each $\theta' \in T(G_c) : \beta^T\theta' \leq b$,
- *tight:* $\beta^T\theta = b$, and
- *max-min complementary:* $\forall l \in \mathcal{K} : (\beta)_l > 0$ if and only if $\theta_l \leq \theta_k$.

The valid inequality $\beta^T\theta' \leq b$ associated with each user k by the above claim will be called the *bottleneck inequality* for k.

Proof. For each $k \in \mathcal{K}$ construct the vector θ' with coordinates defined as follows:

$$\theta'_l = \begin{cases} \theta_l & \text{if } \theta_l \leq \theta_k \\ \theta_k & \text{otherwise} \end{cases}$$

Observe that exactly those users l are non-dominated at θ' for which $\theta_l \leq \theta_k$ and all other users are dominated. Now, simply apply Theorem 2.12 to θ' to obtain a valid inequality that satisfies the claims. □

What is remarkable in this result is that bottleneck inequalities work very much like bottleneck edges in the fixed-path model (hence the name). With this analogy in mind we could rephrase Theorem 2.15 as follows: *an allocation of rates is max-min fair in the generic sense, if and only if all users have a bottleneck (inequality).* This formulation is exactly the same as the one given for the fixed-path model, only the definition of bottlenecks differs somewhat.

Interestingly, the analogy goes even further, since not just bottlenecks but the water-filling algorithm too extends to the general max-min fair allocation problem. Recall that the water-filling algorithm is based on the idea to generate a bottleneck for at least one user in every iteration, no matter in which form bottlenecks are defined. Provided that the bottlenecks arise in the form of a bottleneck inequality, Theorem 2.15 guarantees that what we eventually obtain by running the water-filling algorithm on the throughput polytope is exactly the max-min fair allocation. The below claim formalizes this second important consequence of Theorem 2.15.

Corollary 2.16. The water-filling algorithm is correct to obtain a max-min fair allocation over $T(G_c)$.

Considering the pseudocode given in Algorithm 2.1, it is straightforward to implement this algorithm both in the fixed-path setting and the routing-independent case; hence the name "Generalized water-filling algorithm". The running time of the algorithm then is clearly polynomial provided that the input (the throughput polytope), is of polynomial size; this, however, is not guaranteed in general due to the irrepresentability of the throughput polytope (see Theorem 2.8).

Example 2.9. Consider the sample network of Example 2.5 and execute the water-filling algorithm. As the first step, increase the rate of all users at the same pace, which amounts to, starting from the origin, moving along the direction $[1, 1, 1]$ as long as some users get blocked. This occurs at the point $[\frac{4}{3}, \frac{4}{3}, \frac{4}{3}]$, where the constraint $\theta_1 + 2\theta_3 \leq 4$ becomes active. Fix this constraint as the bottleneck inequality for the blocked users $(1, 6)$ and $(7, 5)$, and increase the rate of the remaining user that is still dominated, $(7, 8)$, as long as possible. This final user gets blocked at the rate of $\frac{5}{3}$ and the resultant allocation, $\theta = [\frac{4}{3}, \frac{5}{3}, \frac{4}{3}]$ is max-min fair. To obtain the bottleneck for the last user, $(7, 8)$, sum the two constraints binding at θ, which yields $\theta_1 + \theta_2 + 3\theta_3 \leq 7$.

The final question that remained to be answered is that, once we computed the max-min fair allocation θ, how to obtain a routing that realizes it. That is, we need to find path-flows $u \in M(G_c)$ so that $\theta_k = 1^T u_k$ for each $k \in \mathcal{K}$. This amounts to solving a multicommodity flow problem (in fact, a simple linear feasibility problem [186]) below with the rate variables fixed at θ:

$$\max\ 0 : \sum_{k \in \mathcal{K}} P_k u_k \leq c \tag{2.14}$$

$$1^T u_k = \theta_l \qquad \forall k \in \mathcal{K} \tag{2.15}$$

$$u \geq 0 \tag{2.16}$$

This can be done in polynomial time [18]. The computed path-flows will then supply a set of forwarding paths and a rate at which users have to distribute their traffic to those paths. In this example we deliberately fixed the objective at zero (see the next section for the motivation); of course operators are free to substitute any objective function they feel important, like minimizing the lengths of the paths, minimizing the maximum link utilization, etc.

Example 2.10. Solving the linear program (2.14)–(2.16) for the sample network of Example 2.5, we obtain that user $(1, 6)$ must split its traffic evenly between the paths $1 \to 2 \to 3 \to 6$ and $1 \to 4 \to 5 \to 6$; user $(7, 8)$ must transmit over the direct link; while user $(7, 5)$ has to apply the traffic splitting ratio $1 : 3$ between the paths $7 \to 8 \to 2 \to 3 \to 4 \to 5$ and $7 \to 8 \to 5$. This routing, once established in the network, will automatically realize the max-min fair throughput allocation θ using the exact same distributed flow control and queuing techniques as in the fixed-path model [132].

2.4.2 Graph-theoretic Interpretation

So far, we have shown how the concept of bottlenecks extend from the fixed-path max-min fair rate allocation problem to the generic case. Analogously to the traditional model, we could obtain an "if and only if" relation between the existence of bottlenecks for each user and max-min fairness, which also guaranteed the correctness of the water-filling algorithm. Regrettably, however, our bottlenecks are currently defined in terms of valid inequalities, which, being more of a geometric concept rather than a network theoretical one, is not really descriptive. Below, we translate this bottleneck argumentation to the more palpable concept of network *cuts* whose properties show remarkable similarity with the properties of "bottleneck edges" in the fixed-path model.

In the heart of the fixed-path model there lies the notion of bottleneck edges. A bottleneck edge is one that blocks any increase in the throughput of the user it belongs to.

This is because *(i)* it is filled up to capacity when we realize the max-min fair allocation, and *(ii)* the corresponding user has the maximum rate among the users that use that edge. This conventional interpretation fails in the generic model, since neither the set of paths nor the users of a particular edge are fixed.

A bottleneck edge blocks one particular, fixed path of some user. To block *all* paths we have to consider an entire set of edges, a *cut*, which, when removed from the network, disconnects every directed path connecting the source to the destination. This suggests the idea to search for the generalization of bottleneck edges in the form of *bottleneck cuts*. What remained to be done is to translate the defining properties of bottleneck edges and bottleneck inequalities to bottleneck cuts.

Let θ be max-min fair in a regular network configuration G_c and choose some user $k \in \mathcal{K}$. Additionally, suppose that we have somehow found the corresponding bottleneck cut $\mathcal{C}_k$ and let $\mathcal{K}_{\mathcal{C}_k} \subseteq \mathcal{K}$ denote the set of users whose source node is separated away from the respective destination node by $\mathcal{C}_k$ (see notations in Table 2.1).

First, consider the following defining property of bottleneck edges in the fixed-path model: "a user's rate is maximal at its bottleneck edge among the users whose path traverses that edge" [107] [26]. In terms of bottleneck cuts, this property will translate into the following: "a user's rate is maximal among the users separated by the corresponding bottleneck cut."

Property 2.17. Given a max-min fair allocation $\theta \in G_c$, for a user $k \in \mathcal{K}$ with bottleneck cut $\mathcal{C}_k$ it holds that

$$l \in \mathcal{K}_{\mathcal{C}_k} \Leftrightarrow \theta_l \leq \theta_k \ .$$

The second defining property of bottleneck edges is that they are always filled to capacity at the max-min fair allocation. Note that in the fixed-path model there is exactly one way to realize a particular rate allocation since the paths and the splitting ratios (in the multipath case) are fixed. In the general model, however, we need to take into consideration *all* routings that can be used to realize the max-min fair rate allocation and we shall require the bottleneck cut to be saturated no matter how, i.e., over which particular sets of paths, we choose to route users' traffic. This leads to the below generalization.

Property 2.18. Given a max-min fair allocation $\theta \in G_c$ and any routing $u \in M(G_c)$ that realizes θ, for a user $k \in \mathcal{K}$ with bottleneck cut $\mathcal{C}_k$ it holds that

$$\forall\ (i,j) \in \mathcal{C}_k : \sum_{l \in \mathcal{K}_{\mathcal{C}_k}} \sum_{P \in \mathcal{P}_l : (i,j) \in P} u_P = c_{ij} \ .$$

Interestingly, these two, fairly naive, generalizations of the defining properties of bottleneck edges establish a sufficiently rich framework for characterizing bottlenecks in the general model. In particular, by Property 2.18 a bottleneck cut of a user k is always saturated, regardless of the chosen routing, and therefore any increase in the rate of user k would decrease the rate of some other user that also traverses the same bottleneck cut but whose throughput is already smaller by Property 2.17. Again, this property is independent of the actual routing. This reasoning then gives rise to the main result of this Chapter, a bottleneck argumentation for the general max-min fair bandwidth allocation problem that is completely analogous to the conventional argumentation for the fixed-path model.

Theorem 2.19. An allocation of rates $\theta \in T(G_c)$ is max-min fair, if and only if each user has a bottleneck cut with Property 2.17 and Property 2.18.

Proof. We have already seen that some $\theta \in T(G_c)$ is max-min fair if and only if each user $k \in \mathcal{K}$ has a bottleneck inequality $\beta^T\theta' \leq b = w^T c$ that is valid, tight at θ, and max-min complementary (see Theorem 2.15). Here, w can be thought of as edge weights and $(\beta)_l$ as the length of the shortest path from s_l to d_l over the weights w (see Proposition 2.9). Now, define the corresponding bottleneck cut for user k as

$$\mathcal{C}_k = \{(i,j) \in E : w_{ij} > 0\} \ . \tag{2.17}$$

Using this setting, $\mathcal{K}_{\mathcal{C}_k} = \{l \in \mathcal{K} : (\beta)_l > 0\} = \{l \in \mathcal{K} : \theta_l \leq \theta_k\}$, where the first equality comes from the observation that $\mathcal{C}_k$ cuts away exactly those users $l \in \mathcal{K}_{\mathcal{C}_k}$ whose shortest path weight, i.e., $(\beta)_l$, is strictly positive, and the second equality follows from max-min complementarity in Theorem 2.15. Thus, $\mathcal{C}_k$ as defined by (2.17) immediately satisfies Property 2.17.

To prove the theorem, we only need to show that it fulfills Property 2.18 too. For this, consider the below primal-dual pair of linear programs, the primal of which we have used earlier to find a routing for a given rate allocation in (2.14)–(2.16) and the dual is simply an optimization problem over the projection cone (2.12)–(2.13):

$$\begin{aligned} &\max\ 0 && && (2.18)\\ &\textstyle\sum_{l\in\mathcal{K}} P_l u_l \leq c && && (2.19)\\ &1^T u_l = \theta_l && \forall l \in \mathcal{K} && (2.20)\\ &u \geq 0 && && (2.21) \end{aligned} \qquad \begin{aligned} &\min\ w^T c - \beta^T\theta && && (2.22)\\ &\textstyle\sum_{(i,j)\in P} w_{ij} \geq \beta_l && \forall l \in \mathcal{K}, \forall P \in \mathcal{P}_l && (2.23)\\ &w^T \geq 0 && && (2.24) \end{aligned}$$

Any routing u that realizes θ is feasible and optimal in the primal (2.18)–(2.21) and the coefficients $[w^T, \beta^T]$ of the bottleneck inequality for user k are feasible in the dual (2.22)–(2.24) and, by tightness $\beta^T\theta = w^T c$, also optimal. Applying complementary slackness from the Karush-Kuhn-Tucker conditions of linear programming [18] to (2.19) and (2.23) we have that

$$(w^T P_l - 1^T \beta_l) u_l = 0 \quad \forall l \in \mathcal{K} \tag{2.25}$$

and

$$w^T (c - \sum_{k\in\mathcal{K}} P_k u_k) = 0 \ . \tag{2.26}$$

Then, for any $l \notin \mathcal{K}_{\mathcal{C}_k}$ we have $\beta_l = 0$ (by again max-min complementarity from Theorem 2.15) and hence $w^T P_l u_l = 0$ by using (2.25), and so we write for each $(i,j) \in E$:

$$w_{ij} > 0 \Rightarrow u_q = 0 \quad \forall l \notin \mathcal{K}_{\mathcal{C}_k}, \forall P_q \in \mathcal{P}_l : (i,j) \in P_q \ . \tag{2.27}$$

Finally, using (2.26) we write

$$\forall (i,j) \in E : w_{ij} > 0 \Rightarrow c_{ij} = \sum_{l\in\mathcal{K}} \sum_{P_q \in \mathcal{P}_l : (i,j)\in P_q} u_q = \sum_{l\in\mathcal{K}_{\mathcal{C}_k}} \sum_{P_q \in \mathcal{P}_l : (i,j)\in P_q} u_q \ , \tag{2.28}$$

where the second equality comes from (2.27). Observing that (2.28) essentially coincides with Property 2.18 completes the proof. □

Example 2.11. Consider again the sample network in Example 2.5. We have seen that users $(1,6)$ and $(7,5)$ get blocked in the first iteration of the water-filling algorithm at the rate allocation $[\frac{4}{3},\frac{4}{3},\frac{4}{3}]$ by the bottleneck inequality $\theta_1 + 2\theta_3 \leq 4$ which is, recall from Section 2.3.1, generated by the weight set $w_{2,3} = w_{4,5} = 1$, $w_{8,5} = 2$, and all zero otherwise. Using (2.17) we get the bottleneck cut $\mathcal{C}_1 = \mathcal{C}_3 = \{(2,3),(4,5),(8,5)\}$. This cut then spectacularly demonstrates the essence of the bottleneck argumentation as of Theorem 2.19, in that it separates away exactly the blocked users $(1,6)$ and $(7,5)$ and it is saturated by any routing that produces the rates $\theta_1 = \theta_3 = \frac{4}{3}$ (in fact, there is only one option to choose from). Finally, the bottleneck cut for the last remaining user, $(7,8)$, is $\mathcal{C}_2 = \{(7,8)\} \cup \mathcal{C}_1$; we kindly encourage the reader to verify that both Property 2.17 and Property 2.18 hold for this cut as well.

As a final remark, we note that the above bottleneck argumentation is valid for any arbitrary regular network, not just the simple and, coincidentally, acyclic ones we cited as examples. Additionally, it is noteworthy to mention that our bottleneck argumentation contains the conventional one as a special case; to see this, it is enough to restrict each user to a single path and observe that bottleneck cuts degrade to the conventional bottleneck edges in this case.

2.5 Related Work

It is difficult to trace back efficient resource allocations and fairness notions to a definite origin; fairness notions find their roots in diverse fields of economy (Atkinson's index [9, 213]), game theory (Nash bargaining [153], Shapley value [188]), sociology [131], political philosophy [111], and even information theory and entropy functions (Rényi entropy [185]). The earliest applications to telecommunications are related to bandwidth allocation problems, see [26, 107, 108, 115, 132]; later this setting has been extended to basically all aspects of computer systems [4, 13, 24, 37, 47, 53, 57, 81, 82, 92, 108–110, 115, 133, 189, 200, 201, 230]. For excellent undergaduate-level text on the subject see [26, 132], and for comprehensive recent surveys consult [110, Appendix D] and [131].

Highlighting its usefulness, several extensions and ramifications of max-min fairness have come to existence throughout the years, like "plain" min-max fairness [133], weighted max-min fairness [26], max-min utility fairness [36], upward max-min fairness [52, 53], α-fairness [115], etc. Note that most of these concepts can be traced back to the unweighted case [133], correspondingly our results generally hold for these extended notions as well. Most existing work is for the fixed-path model; for initial takes on the general max-min fair allocation problem see [150] and [133]. The latter paper by Le Boudec and Radunovic is crucial in the context of this Chapter, as this was the first work that stated bandwidth allocation problems as mathematical programs, gave existence characterizations in terms of the feasible set for these mathematical programs, and called for extending the bottleneck argumentation to the routing-independent, generic case [133, Section "When bottleneck and water-filling become less obvious"]. In this Chapter, we have closed this long-standing open research problem.

Recently, with the trend towards outsourcing web services to large-scale public cloud providers' data centers the problem of fair resource allocation has become immensely more complex, in that now multiple types of resources, like compute [37, 108, 200], storage [13, 110], memory [4], and network facilities [57, 81, 82, 109, 115, 201, 230] all need to be handed out to users *at the same time* and fairness must be guaranteed *simultaneously* across all types of resources. For instance, dominant resource fairness [81] aims for an

allocation that is max-min fair in the dominant (i.e., most intensively used) resource type of each user, multi-resource fairness [109, 110, 230] deals with the case when users can trade-off certain types of resources for others, and constrained max-min fairness [82] develops a fair resource allocation framework for the case when feasible simultaneous allocations of resources are subject to further intricate constraints. Extending our generic formulation and the accompanying treatment to these cases, however, is beyond the scope of this Dissertation and left for further intriguing study herein.

Chapter 3

Adaptive Routing: The Control-theoretic Perspective

TRAFFIC engineering is the art and science of monitoring, analyzing, and optimizing the way traffic is conveyed through a service provider network, in order to deliver the required user experience to customers, to avoid congestion that might cause service disruptions, and to materialize the largest profit margin attainable with the installed network infrastructure [11]. The most important means by which these diverse goals can be realized is a routing algorithm, responsible for mapping traffic demands to the physical network infrastructure. In this Chapter, our focus is the design of such routing and rate-control algorithms.

3.1 Preliminaries

3.1.1 Network Routing and Multipath Rate-control

The main factors to consider in the design of routing and rate-control algorithms is the characteristics of traffic that enters and leaves the network, and the *availability and accuracy of the information on the actual traffic demands* at the time the routing algorithm makes a decision. In cases when the traffic matrix is reasonably static for a longer period of time, historical measurements (and traffic matrices constructed based on them [89, 147]), data mining techniques, and behavioral analyses can be used to make accurate predictions about future demands [225] and provision forwarding paths statically with respect to the predicted traffic characteristics [35, 71]. *Internet traffic*, however, *tends to exhibit substantial variation over a wide range of timescales* due to various reasons beyond the control of the operator [184]. The instabilities and oscillations in the inter-domain routing ecosystem [205,206,208], the emergence of overlay networks and peer-to-peer applications, traffic bursts caused by flash crowds, the emergence of communications protocols without rate-control (e.g., media and VoIP), and the rapidly changing Internet application landscape, are all factors *making accurate traffic matrix estimation and, correspondingly, provisioning static routes increasingly hard* [122, 129, 218].

The problem is that when traffic demands change abruptly on a small timescale the traffic engineering algorithm does not have time to re-adjust static forwarding paths appropriately, leading to congestion, increased packet loss, delay, and jitter, all in all, a deterioration of user experience. Accordingly, traffic engineering algorithms have grad-

ually evolved from what was initially a predominantly static setting [35, 71], through optimizing for multiple traffic matrices [72, 146, 183, 203, 218, 232] towards fully adaptive schemes [25, 70, 77, 98, 112, 116, 117, 130]. Such adaptive routing algorithms make no assumption on input traffic demands whatsoever, but rather try to adapt routing to the temporary demands. In general, the goal of such a *rate-adaptive multipath routing* algorithm is to, given users' traffic demands, distribute these demands dynamically along one or more forwarding paths provisioned for each user in a way so as to minimize congestion and, possibly, fulfill further network-level operational objectives [134].

Rate-adaptive multipath routing is a difficult problem and, correspondingly, prior algorithms proposed in the literature generally adopt various heuristic rate-adaptation schemes that track users' demand dynamics on a "best-effort" basis, with no [25, 70, 77, 98, 112, 116, 117, 130] or only a rather lose characterization on the worst-case congestion that may result in the network [7, 14, 62, 163–165, 212]. In many commercially operated networks, like transit, provider or enterprise networks, often *any* level of congestion is detrimental, given the growing share of inelastic multimedia traffic and the corresponding strict Service Level Agreements (SLAs) and Quality of Service (QoS) requirements posed by the paying customers [70, 98, 112, 218]. So far, it has been an open problem to *find a multipath rate-adaptive routing algorithm that simultaneously achieves (i) provable stability, (ii) optimalilty with respect to any linear or quadratic objective function, and (iii) feasibility*, so that the algorithm can accommodate any admissible traffic matrix in the network without violating link capacities.

3.1.2 Contributions

In this Chapter, we provide such an optimal multipath rate-adaptive scheme. Our main contribution is casting the routing problem in the framework of constrained optimal control theory, which allows us to obtain optimal state feedback routing controllers for any network under mild regularity conditions. Our simulations studies confirm that our routing controllers are viable in small- and middle-sized networks.

The particular contributions are as follows [157, 177, 178]:

- We introduce a control-theoretic model for rate-adaptive multipath routing. Our model allows to design general routing controllers that do not rely on static or estimated traffic matrices but rather dynamically adapt to varying demands.
- We prove that for any network there exists an optimal rate-adaptive multipath routing algorithm that can route any traffic matrix without congestion, provided that each particular traffic matrix would be routable by a static routing that is computed specifically just for that traffic matrix.
- We show that our rate-adaptive multipath routing algorithm is optimal, feasible and stable. In addition, we derive some useful operational properties; e.g., we show that the resultant routing is continuous over the demand set.
- Finally, we give a new complexity characterization for our routing controllers. Our technique is based on a novel use of boundary-triangulations to solve multi-parametric feasibility programs and may be of interest beyond the scope of this Chapter. We show empirically that our technique often gives tighter space characterizations than previous techniques.

We note finally that the models, algorithms, and evaluation results presented in this Chapter constitute just a small fragment of our universal framework rate-adaptive multi-

path routing called *generalized oblivious routing* [158]. In this model, we consider different types of *affine* functions, which embody the very rate-adaptation mechanism in multipath routing, to obtain different routing architectures, be that distributed, centralized, or hybrid. Crucially, this model then lends itself readily to reason about the respective architectures; consequently, for the first time in the literature *we were able to measure different routing architectures against one another on a common ground.* Due to its complexity we will not present the entire framework here; rather we concentrate on just one specific part of the framework, namely, the centralized setting. The interested reader is referred to our comprehensive journal paper [158]; but see also a brief background on generalized oblivious routing in Section 3.5.

The Chapter is organized as follows. In Section 3.2 we recall some notation, we motivate rate-adaptive multipath routing control on an illustrative example, we review the basics of model predictive control (MPC), and we cast rate-adaptive routing in this framework. In Section 3.3 we derive a set of optimal controllers for this system model and discuss important theoretical and operational considerations, and in Section 3.4 we give a new complexity characterization and highlight some preliminary performance evaluation results. Finally, in Section 3.5 we review related work and position our optimal controllers in the general theme of rate-adaptive routing.

3.2 Formal Model

In order to make this Chapter as self-contained as possible, first we briefly recall some notation from the previous Chapter. For a summary of the notations used in this Chapter, the reader is referred back to Table 2.1. Then, we review some important facts from optimal control theory, and finally we present our formal system model for optimal routing control.

3.2.1 Notation

The basic problem of rate-adaptive multipath routing can be formulated as follows. Given a network topology $G(V, E)$ consisting of n nodes and m edges; edge capacities $c = [c_{ij} : (i, j) \in E]$; and a set of source-destination pairs (or users) $(s_k, d_k) \in \mathcal{K}$, each one provisioned[4] a set of paths $\mathcal{P}_k$ and each one presenting its momentary traffic demand θ_k to the network, the task is to adjust sending rates u_P along each path $P \in \mathcal{P}_k$ of each user $k \in \mathcal{K}$ so that no link becomes overloaded (the aggregate flow sent to a link does not exceed the link's capacity). Additionally, one may pose additional constraints on the routing algorithm, like complexity bounds, fairness in allocating network resources, or optimality with respect to some objective function that expresses the performance preferences of the network operator. Below, we deal with the latter case.

The ensuing analysis is built upon the notion of *piecewise affine routing functions*, as they are simple enough to be incorporated into a control-theoretic optimization framework, yet broad enough to express most routing methods relevant to practice, like single-path routing, equal-cost multipath, traffic splitting ratios, etc [30].

Definition 3.1. A piecewise affine (or simply, affine) routing function $\mathcal{S} = \{(\mathcal{S}^i(\theta), \mathcal{R}_i) : i \in \mathcal{I}\}$ is defined as a collection of simple affine functions $\mathcal{S}^i(\theta)$ over a polyhedral partition

[4]Recall, the combination of a particular graph $G(V, E)$, edge capacities c, source-destination pairs $(s_k, d_k) \in \mathcal{K}$, and paths $\mathcal{P}_k$ is called a *network configuration* G_c.

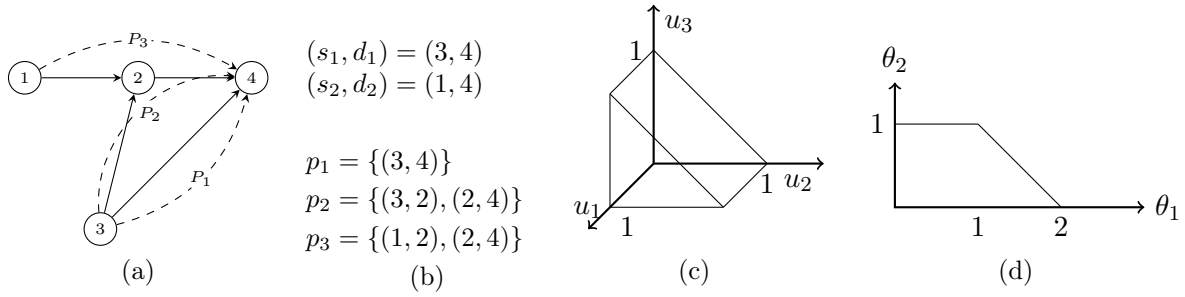


Figure 3.1: A sample configuration: (a) a directed network, (b) source-destination pairs and paths for each user, (c) the corresponding flow polytope and (d) the throughput polytope.

$\{\mathcal{R}_i\}$ of T:

$$\mathcal{S}^i(\theta) = F^i\theta + g^i \text{ whenever } \theta \in \mathcal{R}_i \qquad i \in \mathcal{I} \ ,$$

where F^i are $p \times K$ matrices and g^i are column p-vectors.

Equivalently, when decomposed into separate routing functions $\mathcal{S}_k(\theta)$ for the source-destination pairs $k \in \mathcal{K}$ we get:

$$\mathcal{S}^i_k(\theta) = F^i_k\theta + g^i_k \text{ whenever } \theta \in \mathcal{R}_i \qquad i \in \mathcal{I} \ ,$$

where F^i_k are $p_k \times K$ matrices and g^i_k are column p_k-vectors.

Next, we motivate why piecewise affine routing functions are crucial as the driver for optimal rate-adaptive multipath routing on a simple example.

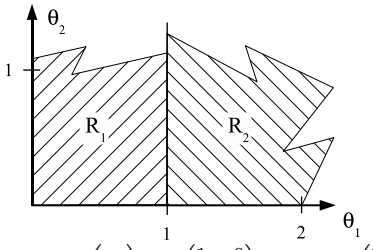
Example 3.1. Consider the simple network depicted in Fig. 3.1. We give two routing controllers for this network in Fig. 3.2a and Fig. 3.2b. Our routing controllers are remarkably simple: they consist of a set of control regions $\mathcal{R}_i$ and the corresponding affine routing functions $S_i(\theta)$, so that the sending rate of users is set to $u = S_i(\theta)$ whenever the traffic matrix θ is in $\mathcal{R}_i$, i.e., $\theta \in \mathcal{R}_i$. For instance, consider the controller in Fig. 3.2b and suppose that both user 1 and user 2 insert 1 unit of traffic into the network. Then, since the traffic matrix $\theta = [1, 1]^T$ is in R_2, we apply routing function $S_2(\theta)$ corresponding to R_2 to obtain the rates $u_1 = 1$, $u_2 = 0$ and $u_3 = 1$. For $\theta = [2, 0]^T$ in the same region we get $u = [1, 1, 0]^T$.

Our sample controller exhibits some appealing properties.

First, the resultant routing is *feasible* in that the sending rate of the users is assigned so that no one link gets over-provisioned no matter what traffic matrix the users present to the network, as long as that traffic matrix is routable with *some* static routing. Recall, the set of "statically routable" traffic matrices θ forms the so called throughput polytope $T(G_c)$ associated with the network configuration G_c (Definition 2.2). Geometrically, feasibility here means that for any $\theta \in T(G_c)$, the corresponding rate-allocation provided by the routing function $\mathcal{S}(\theta)$ is inside the flow polytope $M(G_c)$, that is, $\mathcal{S}(\theta) \in M(G_c)$ (see Definition 2.7). Formally.

Definition 3.2. Given a network configuration G_c, a routing function $\mathcal{S}(\theta)$ is feasible if

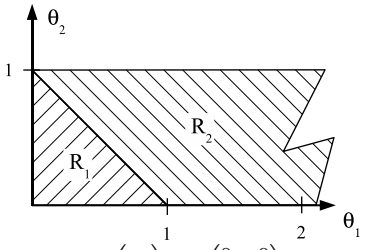
$$\forall \theta \in T(G_c) : \sum_{k \in \mathcal{K}} P_k \mathcal{S}(\theta) \leq c \text{ and } \mathcal{S}(\theta) \geq 0 \ .$$



$$\text{if } \theta_1 \le 1 \text{ then } \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{(R1)}$$

$$\text{if } \theta_1 \ge 1 \text{ then } \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} \quad \text{(R2)}$$

(a) optimal routing controller with control regions(p_1 is cheaper than p_2)



$$\text{if } \theta_1 + \theta_2 \le 1 \text{ then } \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad \text{(R1)}$$

$$\text{if } \theta_1 + \theta_2 \ge 1 \text{ then } \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \quad \text{(R2)}$$

(b) optimal routing controller with control regions (p_2 is cheaper than p_1)

Figure 3.2: Optimal routing controllers and control regions for the sample network in Fig. 3.1: (a) an optimal controller for the case when path p_1 is preferred over p_2, and (b) an optimal controller when path-preference is the other way around. Observe how our controllers can consider operational path-preferences in the routing functions produced.

Second, our sample rate-adaptive routing controllers are *stable* [112, 137]. Stability has multiple connotations in control theory; roughly speaking, stability in the context of rate-adaptive routing means that to any bounded input, i.e., a traffic matrix, a routing function orders bounded output, i.e., a routing (BIBO stability); or that the routing function is such that for any initial state the rates converge to the origin in finite time without permanent oscillations (asymptotic stability). This latter property is crucial in an operational setting and therefore we shall adopt it below. Interestingly, the lack of stability and the resultant network-wide routing oscillations turned out to be the case why the ARPANET was switched from what originally was an adaptive routing scheme, where link costs were varied proportionally to the link load, to a fundamentally static setting [25]. In our case, such oscillations will provably never happen.

Third, our routing controllers are also *optimal* in that the controllers, whenever presented with a choice, will choose a routing that minimizes some cost function that can be freely set by the operator. See formal definitions in [21–23, 28]; below we rather illustrate optimality in routing control with an example.

Example 3.2. Consider the sample network in Fig. 3.1. The routing function in Fig. 3.2a is optimized with respect to a payoff function that assigns smaller cost per unit flow to the single-hop path P_1 than to the two-hop path P_2; i.e., this controller favors minimum hop-count paths. Consequently, the controller fills the shorter path(s) first (e.g., the one-hop path P_1 in Fig. 3.1) and only after the short paths are loaded to capacity it routes additional traffic to the longer path (e.g., the two-hop path P_2 in Fig. 3.1). Contrariwise, the routing controller in Fig.3.2b is optimal for the opposite case, i.e., when the administrative cost of P_2 is smaller than that of P_1 (e.g., because the longer path may still provide smaller delay); in this case the controller first fills P_2 and only after this it fills the less-preferred path P_1, gradually shifting traffic from P_2 to P_1 as user 2 increases its demand to ensure feasibility. Later, we show that the framework allows for more complex objective criteria as well, i.e., quadratic cost functions.

In the rest of this Chapter, we show that such stable, feasible and optimal routing controllers exist in any network, and each one takes the above form: a set of regions and the corresponding affine routing functions. But first we need to recall some basic definitions from optimal routing theory in order to describe rate-adaptive routing in a formal control theoretic model.

3.2.2 Constrained Model Predictive Control

Next, we cast multipath rate-adaptive control in the framework of *(Constrained) Model Predictive Control* (MPC, also called Moving or Receding Horizon Control) [28]. In this framework, a system (or the plant) is described by a model that can be used to predict the dynamics of the system in a given timeframe (control horizon), plus a set of operational constraints that describe admissible states along the system's trajectory. In the usual setting, the system dynamics and the constraints are complex enough to rule out any *offline* solution for the entire time horizon; rather, the control action is obtained by solving *online*, at each time instant, a finite horizon open-loop optimal control problem, using the current state of the plant as the initial state, and applying the first control in the resultant optimal control sequence immediately to the plant. This way, the complexity of the offline solution is significantly reduced and, should the system diverge from the predicted trajectory due to, e.g., an unexpected input or inaccuracies in the model, the controller can still adapt by being reinitialized from the current state in each time step. At the same time, this online approach requires solving a mathematical program in each time step, which may be prohibitive for systems with very fast dynamics, like rate-adaptive routing.

Nevertheless, for the simplest form of MPC, linear MPC, a fast offline solution can often be obtained with reasonable computational effort. This reduces the online control computation to the simple evaluation of an explicitly defined (closed form) piecewise linear function, which allows to effectively regulate even extremely dynamic plants. The price we pay for the simplicity of offline MPC is skyrocketing space (storage) complexity for the resultant controller, but for smaller systems where space complexity is not that important offline control is undoubtedly the preferred choice over online control (see below for space complexity characterizations). Below, we shall adopt this offline approach for rate-adaptive routing.

Suppose that a system is characterized by some *state* (x), *input* (u) and *output* (y) variables, whose evolution in time is governed by the following general linear system [195]:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \\ x(0) &= x_0 \end{aligned} \tag{S}$$

Here, A, B, C, and D are constant matrices of proper size and $x(t)$, $u(t)$ and $y(t)$ are the values of the state(s), input(s) and output(s), respectively, at time t. Additionally, a set of constraints can be specified to which the system state and the input must obey at every time instance; e.g., the framework admits polyhedral constraints $Qx(t) \leq q$.

Suppose, in addition, that we are given an objective function, the cost function or the *payoff* function, which prizes the evolution of the system in time as the function of the

input and the initial state:

$$P(u(.), x(0)) = q_f^T x(N) + \sum_{t=0}^{N-1} (r^T u(t) + q^T x(t)) \ , \tag{P}$$

where N is the *control horizon* and q_f^T, r^T and q^T (all constant row-vectors of proper size) are the *terminal cost* and *running payoffs* for the input and the state, respectively. For completeness, we note that the framework allows for linear [21, 22] as well as quadratic payoffs [23]; below, we concentrate on the linear setting for brevity.

Now, the basic problem of optimal control theory is to design a controller to adjust the input u, so that the system (S), starting from some initial state $x(0)$, is regulated obeying the constraints (C) along an optimal trajectory, as measured by the payoff function (P). In this setting, u is called an *optimal control*.

3.2.3 Optimal Rate-adaptive Routing Control

Next, we cast multipath rate-adaptive routing in the above control-theoretic model.

In our model, the system state is the amount of traffic waiting to be served at the source nodes, the output is simply this same state (which therefore we shall omit henceforth), and the control is the amount of traffic placed at individual paths of the users. Formally, let $x(t)$ be a column K-vector whose kth component describes the amount of data to be delivered from s_k at time t, and let $u_P(t)$ describe the flow routed at path $P \in \mathcal{P}_k, k \in \mathcal{K}$ at t.

Given the initial conditions $x_k(0) = \theta_k : \forall k \in \mathcal{K}$, our model will be characterized by the following system dynamics:

$$x_k(t+1) = x_k(t) - \tau \sum_{P \in \mathcal{P}_k} u_P(t) \qquad \forall k \in \mathcal{K} \tag{D}$$

$$x_k(0) = \theta_k \qquad \forall k \in \mathcal{K} \tag{I}$$

The state $x(t)$ integrates the data fed by the users at the source nodes into the network at time zero (the initial state), minus the sum of flows carried away along the individual paths of the user within the discrete time step τ. In other words, $x_k(t)$ models the amount of traffic accumulated in the input buffer of each source-destination pair k at time t, and the initial state $x_k(0)$ is simply the demand of user k presented to the system at the zero-th time instance. For the sake of simplicity, we shall assume henceforth that the discrete time step is 1 unit and θ is scaled accordingly, and so we shall omit τ in the equations. In the above model we assume that no further traffic arrives within the time frame τN; this assumption will be relaxed later by adopting a 1-step receding horizon control model.

The control must respect certain operational constraints in assigning rates to the users; in particular, edge capacities may not be violated at any instance of time:

$$\sum_{k \in \mathcal{K}} P_k u_k(t) \leq c \ ; \tag{C1}$$

rates are non-negative:

$$\forall k \in \mathcal{K} : \quad u_k(t) \geq 0 \ ; \tag{C2}$$

and the controller cannot clear more data from the source nodes than it is available there:

$$\forall k \in \mathcal{K} : \quad x_k(t) \geq 0 \ . \tag{C3}$$

Observe that (C1) and (C2) coincide with the feasibility constraint of Definition 3.2 in that they require $u(t)$ to be a valid routing; or in other words, we require $u(t) \in M(G_c)$ in every time step. Let (C) denote the full constraint system (C1)–(C2)–(C3).

Finally, the objective of the routing controller is to minimize the cost of the resultant routing over the time horizon N or, equivalently, to maximize the payoff function:

$$\min P(u(.), x(0)) = q_f^T x(N) + \sum_{t=0}^{N-1} r^T u(t) + q^T x(t) \ . \tag{P}$$

In this formulation, setting the terminal cost $q_f^T > 0$ will drive the system to eventually settle in, or as close as possible to, the origin; other formulations may enforce any desired terminal state T_f by adding an explicit constraint $x(N) \in T_f$.

Given a network configuration G_c, the dynamics (D), the initial condition (I), the constraints (C), and the payoff function (P) together make up the *Optimal Rate-adaptive Routing (ORAR)* model for G_c.

3.3 Optimal Controller Design

Next, we design an optimal controller for the ORAR model described above. The controller's job will be to remove as much data from the inputs as possible; in other words, the controller regulates the states towards the origin. The main contribution in this Section is summarized by the below result.

Theorem 3.3. Given a regular network configuration, there is an offline one-step receding horizon controller, called the *ORAR controller*, that is feasible, optimal, and stable under the ORAR model for any initial state $\theta \in T(G_c)$.

The rest of this Section is devoted to prove the above claim. The proof will consist of a series of technical results, which together will support Theorem 3.3.

The first step of controller design is to convince ourselves that our system is well-behaved and so a suitable controller exists. We say that a system is *controllable* if there exists a control that drives it from any optional initial state into the origin in finite time, and it is *observable* if it is possible to identify the state of a system at any instance of time through output measurements.

Lemma 3.4. If a network configuration G_c is regular and $q_f^T > 0$, $r^T \geq 0$, and $q^T \geq 0$, then the system is both controllable and observable under the ORAR model.

Recall, a regular network configuration G_c according to Definition 2.1 is such that there is a path in G_c from s_k to d_k for each user $k \in \mathcal{K}$ and all edge capacities are finite and strictly positive. Regularity was key to our fairness notions in the previous Chapter; in this Chapter we reuse the same mild regularity conditions to characterize network configurations that admit an optimal routing scheme.

Proof. First, observability is trivial since in the ORAR model the output corresponds to the system state so we can measure it directly. Controllability, furthermore, is also easy to show. Consider a trivial controller that puts some small nonzero flow to the usable paths of each user in each time step, subject to (C); such a trivial controller always exists in a regular network and gradually clears all data from each source node, driving the system to the origin in finite time. □

We note that the result is only sufficient and not necessary; see [23, 28] for stronger characterizations.

The second result concerns the existence of the ORAR controller.

Lemma 3.5. Given a regular network configuration G_c, consider the ORAR model defined by the plant

$$x_k(t+1) = x_k(t) - \tau \sum_{P \in \mathcal{P}_k} u_P(t) \qquad \forall k \in \mathcal{K} \qquad \text{(D)}$$

$$x_k(0) = \theta_k \qquad \forall k \in \mathcal{K} \qquad \text{(I)}$$

the constraints

$$\sum_{k \in \mathcal{K}} P_k u_k(t) \leq c \qquad \text{(C1)}$$

$$u_k(t) \geq 0 \qquad \forall k \in \mathcal{K} \qquad \text{(C2)}$$

$$x_k(t) \geq 0 \qquad \forall k \in \mathcal{K} \qquad \text{(C3)}$$

and the payoff

$$\min P(u(.), x(0)) = q_f^T x(N) + \sum_{t=0}^{N-1} r^T u(t) + q^T x(t) \qquad \text{(P)}$$

where $q_f^T > 0$, $r^T \geq 0$ and $q^T \geq 0$. Then, for any $N > 0$ there exists a control law that, starting from any initial state $\theta = [\theta_k : k \in \mathcal{K}]$ (I), regulates the network according to the dynamics (D), satisfying conditions (C), and optimizing the payoff function (P).

Proof. Consider the below linear program that describes the ORAR model:

$$\min \quad q_f^T x(N) + \sum_{t=0}^{N-1} r^T u(t) + q^T x(t) \qquad \text{(P)}$$

$$\text{s.t.} \quad x_k(t+1) = x_k(t) - \sum_{P \in \mathcal{P}_k} u_P(t) \qquad \forall k \in \mathcal{K}, \forall t \in \{0, \ldots, N-1\} \qquad \text{(D)}$$

$$x_k(0) = \theta_k \qquad \forall k \in \mathcal{K} \qquad \text{(I)}$$

$$\sum_{k \in \mathcal{K}} P_k u_k(t) \leq c \qquad \forall t \in \{0, \ldots, N-1\} \qquad \text{(C1)}$$

$$u(t) \geq 0 \qquad \forall t \in \{0, \ldots, N-1\} \qquad \text{(C2)}$$

$$x(t) \geq 0 \qquad \forall t \in \{1, \ldots, N\} \qquad \text{(C3)}$$

The size of this linear program is polynomial in the size of G_c and the control horizon N. In fact, an online MPC controller would solve exactly this linear program in each time step, setting the initial state θ_k from the current state of the system. To get an *offline* controller, we can solve this system as a multi-parametric linear program, using e.g., the algorithm in [30], to obtain a control law that is general in the initial state parameter θ_k; i.e., the initial state is not fixed but rather it is revealed to the controller during runtime. The result is a function $u(.) = \mathcal{S}(\theta)$, which to each initial state θ orders a routing action $u(t)$ at each time instance t throughout the control horizon. Then, the existence of such a solution as the function of the multi-dimensional parameter θ is guaranteed by [23, Corollary 2]. □

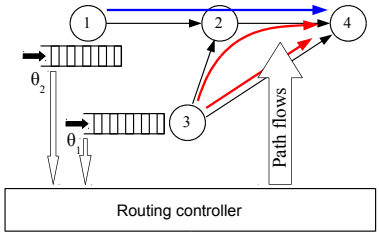


Figure 3.3: The centralized ORAR routing controller architecture.

Suppose $N = 1$ and call the control law $u(\theta)$ obtained by solving the above multi-parametric program the ORAR control law. See Example 3.1 for the sample ORAR controllers.

Next, we show a set of useful properties of the ORAR control and the corresponding payoff dynamics (the so called *value function*). We will generally omit the proofs; the reader is referred to the manuscripts [23,84,85,167] for a general theory of optimal offline model predictive control of constrained systems and the related stability analysis [28]; the below results are applications of the results therein. For full reference, the detailed proofs can be found in our earlier work [158,177].

Lemma 3.6. The ORAR control law $u(.)$ is a continuous and piecewise affine function of θ:

$$u(\theta) = F_i\theta + g_i \qquad \text{if } \theta \in \mathcal{R}_i, i = 1, \dots, r \ ,$$

where $\mathcal{R}_i$s are closed polyhedral sets in $\mathbb{R}^K$. Alternatively, if the ORAR system is solved for an initial set state T_0, then the control regions $\mathcal{R}_i$ partition T_0.

Lemma 3.7. The ORAR control law $u(.)$ is asymptotically stable.

Lemma 3.8. The ORAR control optimizes any linear payoff whenever $q_f^T > 0$, $r^T \geq 0$ and $q^T \geq 0$ and the value function is continuous, convex, and piecewise affine.

Lemma 3.9. The set of initial states for which the ORAR controller converges in 1 steps to the origin (the 1-step feasible set) equals $T(G_c)$.

The significance of Theorem 3.3 is that, theoretically, no information on expected traffic is necessary to design a multipath rate-control algorithm that guarantees feasibility, stability, and optimality over the entire demand space $T(G_c)$. The controller will clear out any traffic demand θ from the source nodes in a single step as long as $\theta \in T(G_c)$; this can be easily seen by observing that the solution to the multi-parametric linear program for a given θ (the offline ORAR control) is *exactly the same* as the optimal solution of the (non-parametric) linear program where θ is fixed (the online ORAR control), and the set of θ for which the fixed linear program is feasible is, by definition, $T(G_c)$. In other words, our offline controller assigns a feasible routing to any traffic matrix for which an online controller would produce a feasible static routing, but instead of having to solve the linear program (P)–(I)–(D)–(C) online in each timestamp, our controller can be *precomputed* offline, still guaranteeing convergence to the origin in a single timestep ($N = 1$).

In operation, the ORAR controller node periodically scans the network, reads the momentary traffic demands θ from the sources, solves a series of polyhedron inclusion problems to find $i \in \mathcal{I}$ so that $\theta \in \mathcal{R}_i$, evaluates $u = F_i\theta + g_i$ to find the optimal routing

for θ, and downloads the resultant traffic splitting ratios to the routers. See Fig. 3.3 for a schematic model of this architecture. This scheme fits perfectly into the centralized control framework advocated for Software Defined Networks (SDN), which makes the ORAR controller an appealing candidate for SDN traffic engineering [3].

An additional benefit is that *the ORAR controller allows for optimizing the routing function through specifying the objective (P)*; both linear and convex quadratic objective functions are permitted [23,28,30]. Plausible objectives would be to minimize delay or the maximum link utilization. Furthermore, *continuity of the control law, both inside routing domains an across boundaries, guarantees smooth routing transients.*

3.4 Complexity

To obtain the ORAR control law, one needs to solve a multi-parametric linear or quadratic program, which is, although computationally quite involving, viable thanks to recent advances in geometric multi-parametric programming [23, 30]. Unfortunately, the resultant control law may prove prohibitively complex, as there is no polynomial upper bound on the number of control regions and individual simple routing functions that emerge when solving the multi-parametric linear program [23, 30]. When $\mathcal{I}$ exceeds about 10^5, centralized routing becomes impractical as the controller spends most of its time solving polyhedron inclusion problems trying to figure out which individual routing function to apply. Storage requirements too can become an issue.

Consequently, the number of control regions in the control law chiefly determines the practical applicability of the ORAR controller. Unfortunately, the worst-case complexity bound provided in [23, Section 4.4] is somewhat lose and largely prohibitive in practice. The next result, which may be of interest beyond the context of ORAR, rather bounds the complexity of 1-step ORAR control to the size of the smallest boundary-triangulation of the throughput polytope (recall from Definition 2.3 that a boundary-triangulation is a partition of a polytope into a set of simplices so that all vertices of the simplices lie on the boundary of the polytope). Complexity of boundary-triangulations is a heavily researched area [20]; consequently our bound often provides tighter complexity characterizations than the worst-case result in [23]. Unfortunately, we pay a price for reduced size; namely, our triangulation-based scheme relaxes the optimality requirement in ORAR control and it focuses on feasibilty instead. Such problems with an empty payoff function are called *multi-parametric feasibility problems.*

The main result is as follows.

Theorem 3.10. Given a regular network configuration G_c, consider the ORAR model with an empty payoff, described by the dynamics (D), the initial condition (I), the constraints (C), let $N = 1$ and add the terminal condition

$$x_k(1) = 0 \qquad \forall k \in \mathcal{K} \ . \tag{T}$$

Then, the minimal number of control regions in the 1-step ORAR controller for the system (D)–(I)–(C)–(T) is upper bounded by the size of the minimal boundary-triangulation of $T(G_c)$.

The Theorem is the corollary of the following claim.

Lemma 3.11. For a regular network configuration G_c and any boundary-triangulation $Q_i : i \in \{1, \dots, q\}$ of T, there exists a continuous compound affine routing function $\mathcal{S} = \{(\mathcal{R}_i, \mathcal{S}_i) : i \in \mathcal{I}\}$ so that $\mathcal{I} = \{1, \dots, q\}$, $\mathcal{R}_i = Q_i$, and $\forall i \in \mathcal{I}, \forall \theta \in \mathcal{R}_i : \mathcal{S}_i(\theta) \in M(G_c)$.

Proof. Let $\theta_1, \ldots, \theta_s$ be the extreme points of $T(G_c)$ and let $u_1, \ldots, u_s$ be path-flows so that $\mathcal{T}(u_i) = \theta_i$ for all $i \in \{1, \ldots, s\}$. (Recall, $\mathcal{T}$ is the throughput mapping as per Definition 2.4: $\mathcal{T}(u) = [\theta_k = 1^T u_k : k \in \mathcal{K}]$). For any simplex Q_j in the boundary-triangulation of $T(G_c)$, $Q_j = \mathrm{Conv}\{\theta_{i_0}, \theta_{i_1}, \ldots, \theta_{i_K}\}$ for some $K+1$ affinely independent extreme points of $T(G_c)$; let $u_{i_0}, \ldots, u_{i_K}$ be the path-flows corresponding to the extreme points of Q_j: $\theta_{i_0}, \ldots, \theta_{i_K}$. Without loss of generality, choose θ_{i_0} as a basis point and define the $K \times K$ matrix $B_j = [\theta_{i_1} - \theta_{i_0}, \ldots, \theta_{i_K} - \theta_{i_0}]$. Note that B_j is invertible. Let u_{i_0} be the path-flow realizing θ_{i_0} and U_j be a $p \times K$ matrix defined as $U_j = [u_{i_1} - u_{i_0}, \ldots, u_{i_K} - u_{i_0}]$.

Consider an arbitrary point $\theta \in Q_j$. Then, there exist $\lambda_0, \lambda_1, \ldots, \lambda_K$ with $\lambda_k \geq 0$ and $\sum_{k=0}^{K} \lambda_k = 1$, so that

$$\begin{aligned}\theta = \sum_{k=0}^{K} \lambda_k \theta_{i_k} &= \sum_{k=0}^{K} \lambda_k \theta_{i_k} - \theta_{i_0} + \theta_{i_0} \\ &= \sum_{k=0}^{K} \lambda_k \theta_{i_k} - \sum_{k=0}^{K} \lambda_k \theta_{i_0} + \theta_{i_0} = \sum_{k=0}^{K} \lambda_k (\theta_{i_k} - \theta_{i_0}) + \theta_{i_0} \\ &= \sum_{k=1}^{K} \lambda_k (\theta_{i_k} - \theta_{i_0}) + \theta_{i_0} = B_j \lambda + \theta_{i_0} \ , \end{aligned} \tag{3.1}$$

where λ is a column K-vector formed by the coordinates λ_k. Consider the path-flow

$$u = \sum_{k=0}^{K} \lambda_k u_{i_k} = \sum_{k=1}^{K} \lambda_k (u_{i_k} - u_{i_0}) + u_{i_0} = U_j \lambda + u_{i_0} \ . \tag{3.2}$$

Observe that u is a routing for θ, as $\mathcal{T}(u) = \mathcal{T}(U\lambda + u_{i_0}) = \mathcal{T}(U)\lambda + \mathcal{T}(u_{i_0}) = B\lambda + \theta_{i_0} = \theta$. In addition, $u \in M(G_c)$ as u is a convex combination of vectors in $M(G_c)$. Noting that $\lambda = B_j^{-1}(\theta - \theta_{i_0})$ by (3.1), we have that

$$u = U_j B_j^{-1}(\theta - \theta_{i_0}) + u_{i_0} = U_j B_j^{-1} \theta + (u_{i_0} - U_j B_j^{-1} \theta_{i_0}) \tag{3.3}$$

is a feasible affine routing function for θ. Since the above holds for any $\theta \in Q$ we conclude that (3.3) is a routing function on the entire simplex Q_j. Using the above construction on each $Q_i : i \in \{1, \ldots, q\}$ gives a piecewise affine routing function $\mathcal{S} = \{(\mathcal{S}_j, Q_j) : j \in \{1, \ldots, q\}\}$. Finally, continuity is trivial by (3.1) and (3.2). □

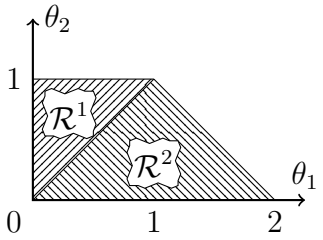
Note that finding a triangulation for which q is minimal is a very difficult computational problem [20] and even if we manage to find one, the size q may still be exponential. We found that in practice the ORAR controllers produced by boundary-triangulations are often less complex, especially for smaller systems where obtaining an optimal (minimal) boundary-triangulation is computationally feasible.

Example 3.3. For the running example, a minimal boundary-triangulation of $T(G_c)$ and the corresponding piecewise affine ORAR routing function is given in Fig. 3.4.

A straightforward way to further reduce controller complexity would be to increase the control horizon N. Recall that N connotes the time the controller is allowed to spend driving the system into the origin. Thus, the larger the control horizon the slower the controller. This is expected to yield larger control regions and hence to decrease

$$\mathcal{S}_1 : \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$$
$$\mathcal{S}_2 : \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}$$

(a)



(b)

Figure 3.4: The ORAR routing function and the corresponding boundary-triangulation.

complexity. It must be noted, however, that the ORAR model, in its current form, assumes that no further traffic arrives into the input buffers within the time frame τN. When this assumption is violated, the system state as predicted by the controller and the real system state diverge. This problem emerges only for N-step controllers where $N > 1$; for $N = 1$, no prediction is needed as the system is cleared out in a single step. And even when $N > 1$ a moderate model inaccuracy does not pose problem in receding horizon control, since we consider only the first control action that is based on exact system state (the initial state).

Setting the control horizon not only affects controller complexity, but it also has profound impact on the set of states to which the controller orders control action. In general, the N-step feasible set, the set of states from which the ORAR plant converges into the origin in N steps, monotonically increases with N and it precisely coincides with the set of states to which an N-step ORAR control orders control action. An 1-step ORAR controller covers only the set of admissible traffic matrices $T(G_c)$, and increasing the control horizon monotonically broadens the range of traffic matrices the ORAR controller can handle.

Fig. 3.5 gives a quick roundup on the practical complexity or ORAR controllers; for detailed numerical evaluation see [157, 158, 177, 178]. The results are presented for a well-known network topology that is often considered in the related literature as a representative ISP topology: the NSFNET Phase II network [43] consists of 12 routers and 128 links. We generated increasingly complex scenarios by increasing the number of users K. The source-destination pairs for each K were chosen according to the bimodal distribution and 2 maximally node-disjoint paths were provisioned per user [89, 147].

Fig. 3.5 gives the average number of control regions for the ORAR controller when K is varied between 1 and 9. Recall, this controller is provably stable, feasible, and optimal, therefore the most important question in this case is the price we pay in terms of complexity for optimality. We conclude that *ORAR control is viable only for networks serving only a couple of users*, but complexity quickly becomes prohibitive when the total number of paths in the system surpasses about 20. To answer this challenge, in our later work [157,158,177,178] we present a family of distributed, centralized, and hybrid routing controllers, each striking a different balance between complexity and performance.

3.5 Related Work

In this Section, we position our results in the context of recent work on traffic engineering and multipath rate-control algorithms.

Traffic management algorithms strive to eliminate network-wide congestion. To this

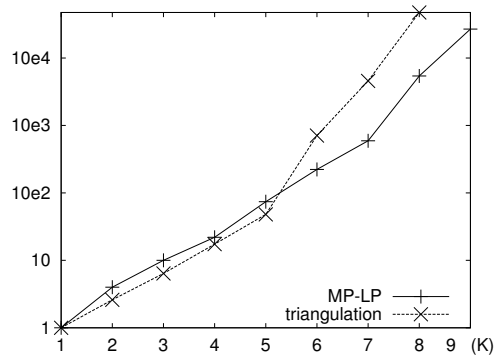


Figure 3.5: The complexity of the ORAR controllers on the NSFNET Phase II network topology [43]: the number of control regions obtained with the multi-parametric programming method ("MP-LP") and from a random (not necessarily minimal!) boundary-triangulation ("triangulation"), as the function of the number of users K. Note the logarithmic scale on the y axis.

end, a *flow-control algorithm* adapts the rate at which source nodes emit traffic in concert with the congestion feedback received from the network. Examples are various versions of the venerable Transmission Control Protocol (TCP) and other network utility maximization schemes [41]. Recently, there has been a trend towards generalizing these flow-control algorithms to the case when users communicate over more than one path and they actively control not only the source rate but the fraction of traffic routed along the individual paths, or the paths themselves, as well [94, 98, 99, 116, 117, 229]. This brings us to the second form of traffic management algorithms: *multipath rate-control* [70,77,98,99,112,129,224]. Here, the amount of traffic to be routed is given and the task is to allocate the load on the forwarding paths in a way to minimize congestion. While flow control is chiefly an end-to-end scheme, multipath rate-control is much better suited to an intra-domain traffic engineering setting, where the domain border routers do not have control over the sending rate of each individual endhost.

A simple form of rate-adaptation is to periodically recompute routers' forwarding tables with respect to fresh measurement data. Basically any static scheme [35, 71, 147] can be made adaptive this way. It is an untrivial task, however, to decide when to recompute: intervening too often causes frequent traffic fluctuations, instability, and adverse packet reordering, while re-computing too rarely might lead to traffic loss as the traffic matrix and the routing might get out of alignment [146,147]. Promising ways to overcome this problem is to design a routing that is "sufficiently good" for multiple traffic matrices at the same time [72,203,218,232], or to make no assumption on input traffic whatsoever. This extreme setting is called *oblivious routing* [7, 25, 70, 77, 98, 112, 116, 117, 130, 163].

Right from the beginning, multipath rate-control algorithms were conceived to be distributed [77], which means that only information available to a router locally can be used to make routing decisions. It quickly turned out, however, that conflicting decisions made by routers unaware of each other's state can easily lead to wide-scale route oscillations [25, 118]. A minimalistic approach to eliminate instability is to apply no rate-adaptation at all: e.g., in oblivious routing traffic splitting ratios are set statically to minimize congestion over *any* combination of demands [7, 14, 62, 163–165, 212]. Curiously, oblivious routing can be surprisingly efficient: in his seminal work, Räcke showed that in undirected graphs we pay only a polylogarithmic factor in congestion compared to the best attainable routing [163]. He later improved the worst-case bound to purely logarith-

mic [164], which is asymptotically tight [17]. Directed graphs do not admit a logarithmic upper bound [14], but in most cases relevant to practice the congestion penalty reaches at most 2 [7]. Even though surprisingly small, this bound still allows substantial link over-subscription [7, 218].

It seems, therefore, that some forms of rate-adaptation is inevitable, but special care must be taken to avoid instabilities and wide-scale route oscillations [25, 118]. Most proposals, therefore, introduce some forms of a signaling mechanism to collect state information from the network. TeXCP applies periodic path probing to collect link utilization information [112], REPLEX uses a complete distance-vector protocol infrastructure to distribute the network state [70], while DATE and TRUMP rely on timely feedback from the network [98,99]. This leads to control overhead, hampers implementation and deployment, and often causes sub-optimality when links are not allowed to be saturated to full capacity to avoid instability [98, 99]. For a promising approach to mitigate these issues, see [129]. As far as we know, the centralized ORAR control framework presented in this Chapter is the first multipath rate-control algorithm that combines feasibility, in that it provably avoids congestion, stability, in that there can be no route oscillations [25, 77], and optimality, in that it allows to optimize any payoff, in a practical and mathematically sound routing scheme.

Feasibility, stability and optimality of ORAR control, however, comes at a distinct price: moving from the distributed model to a centralized one. In ORAR control routers merely execute the forwarding commands sent by the controller [178], which makes ORAR control an appealing candidate for implementing traffic engineering in software-defined networks [144, 226]. Unfortunately, a centralized architecture presents its own set of challenges: the central controller is a single point of failure and scales poorly, and it also needs a stable information exchange mechanism implemented by all nodes. This, in turn, may induce dead-time control instabilities due to the delay between when data is measured and respective control action is taken.

Recently, there has been a trend towards a *hybridization* of routing architectures, where decision making is partially migrated to a central controller. A good example is [224], where a central node computes and sets link weights based on which routers can calculate the best traffic splitting ratios independently. In [177], we present a hybrid oblivious routing scheme that fits neatly into this trend. The numerical evaluations suggest that our hybrid scheme introduces very little added complexity compared to fully-distributed schemes but, unfortunately, the provable optimality and stability of the centralized ORAR scheme is now lost.

It seems that each of the architectural models of distributed (e.g., oblivious routing), centralized (as presented in this Chapter), and hybrid multipath rate-control (as of our prior work, [177]) bring about their own benefits and pose their own challenges. Measuring these against each other so far has been possible on a purely ad-hoc basis, supported only by piecemeal analysis and anecdotal evidence, but an all-encompassing mathematical framework has been missing. In order to close this gap, in our recent work *we propose a new model for rate-adaptive multipath routing that we call generalized oblivious routing, which allows one to analyze distributed, centralized, and hybrid routing architectures within a single framework*, and to develop quantitative as well as qualitative arguments regarding their optimality, stability, and realizability [157, 158, 177]. Our framework is a novel generalization of "conventional" oblivious routing; in particular, the oblivious routing scheme as per [7, 14, 62, 163–165] arises as the *distributed* realization of our model, the

ORAR control model presented in this Chapter boils down to the *centralized* incarnation, and our hybrid distributed-centralized algorithm as of [177] provides the middle-ground, combining the simplicity of the distributed realization with the efficiency of the centralized one. In this regard, our work can be seen as a sequel to [41]: whereas Chiang *et al.* in [41] provide the first comprehensive mathematical framework to understand *control function layering* in network architectures, ours is the first mathematical framework for understanding the *organization of control* in network architectures, be that distributed, centralized, or hybrid.

Chapter 4

Scalable Internet Routing: The Algebraic Perspective

The main concern in the previous Chapters has been the efficient and fair mapping of users' requested traffic demands to the underlying network topology. This task arises most prominently in an *intra-domain* setting, a fixed scope domain in the Internet architecture where the forwarding paths, the bandwidth allocation, and multipath rate-control are all tightly controlled by a single network administrator. In the *inter-domain* setting, however, the Internet consists of thousands network domains operated autonomously by independent service providers. Consequently, there is only very loose control any single stake-holder (i.e., operator) can exert over the end-to-end path, the bandwidth, and the overall traffic rate through the potentially dozens of Internet administrative domains user traffic traverses (although, strikingly, end-to-end rate-control algorithms can still provide certain efficiency and fairness guarantees even at this vast scale [41, 110]).

In this and the subsequent Chapters, *our attention moves to the most critical aspect of large-scale network routing: scalability* [126]. In particular, we aim *(i)* to define scalable routing schemes that minimize the amount of routing information that needs to be stored at, and synchronized across, network nodes, but still retain our capability to define certain network-level performance or service objectives for the resultant routes (this Chapter), and *(ii)* to study routing-state-compression mechanisms that allow routers to compress the routing information they have to maintain to the minimum theoretically possible size (next Chapter). First, in this Chapter we shall extend the theory of compact routing from shortest-path routing to essentially arbitrary routing policies, which will allow us to get a more meaningful characterization of the memory requirements of Internet routing which, contrary to general belief, is *not* shortest-path-based.

4.1 Preliminaries

4.1.1 Compact Routing and Routing Policies

Compact routing theory is the research field aimed at identifying the fundamental scaling limits of shortest-path routing and constructing algorithms that meet these limits [49, 73, 78–80, 126, 209]. Shortest-path routing is a key ingredient in many modern network architectures, as it generally ensures low transmission delay while also minimizes the effort needed to transmit one unit of information from the source to the destination. To

what extent shortest-path routing can scale to large networks, in terms of the memory requirements of implementing the local forwarding functionality at network nodes, has been on the research agenda for a long time.

It has been shown that, in general, it is impossible to implement shortest-path routing with routing tables whose size in all network topologies grows slower than *linear* with the increase of the network size [73, 80]. This negative result paints a rather pessimistic picture on the viability of large-scale Internet routing: in terms of this result memory in Internet routers' data plane needs to be upgraded with adding a constant number of storage bits for every node introduced into the network. Network operators would rather opt for a *logarithmically scaling* memory footprint, where a constant number of additional bits is enough to handle even the doubling of the network size.

To answer this challenge, *compact routing research seeks algorithms to decrease routing table sizes at the price of letting packets to be routed along slightly suboptimal paths*. In this context, "slightly suboptimal" means that the forwarding paths are allowed to be longer than the shortest ones, but length increase must be bounded by a constant *stretch* factor. By now, the research community has built a strong theoretical foundation for compact shortest-path routing, fully characterizing the stretch and memory footprint on a broad catalog of network topologies including hypercubes, trees, scale-free networks, and planar graphs [74, 75, 78, 125, 209], and generic network topologies [49, 209].

It turns out, however, that shortest-path routing is a rather poor description of the path selection preferences (the so called *routing policies*) in use over the Internet. This is because *operators usually consider a broader set of attributes beyond mere path length when provisioning routes*, in order to ensure an expedient flow of user traffic through the network. Such additional attributes may include path reliability and resilience constraints [228]; bandwidth and perceived congestion [6,140,219]; business relations and service level agreements between Internet service providers [5, 33]; security, etc. These path selection strategies are usually described under the umbrella of policy routing. Practically speaking, a *routing policy* is a function that selects a preferred transmission route from the set of all forwarding paths available between two endpoints, according to predefined requirements.

Indeed, a significant portion of the Internet today runs over policy routing [10, 33, 135, 219, 228]. However, currently theoretical scalability characterizations are available only for shortest-path routing, which leaves a considerable gap in our understanding of the long term sustainability of the Internet. The challenge we tackle in this Chapter is to take the first steps towards filling this gap, by defining a mathematical framework to characterize the scalabiliy of routing policies beyond mere shortest-paths.

4.1.2 Contributions

In this Chapter, we construct a mathematical model what we call algebraic compact (policy) routing. This model allows us to study the memory requirements of routing policies in a simple algebraic model and identify the fundamental scalability characteristics of general policy routing. In doing so we build on the recent work of Sobrinho and Griffin [86, 91, 191, 192], which describes disparate routing policy structures in a single theoretical framework using the notion of *routing algebras*. A routing algebra abstracts away the syntactic and semantic diversity of routing policies and, consequently, it lets us to study routing policies in a general, abstract sense. Using this framework, we give an algebraic characterization of the scalability of policy routing and we take a look at the applicability of *constant-stretch compact routing schemes* in an abstract algebraic setting.

Our contributions in particular are as follows [88, 175, 176, 236].

- We introduce *algebraic compact routing*, an extension of the compact routing model defined by Fraigniaud and Gavoille [73, 80] from shortest-path routing to practically arbitrary routing polices, by taking a novel abstract algebraic approach.
- We identify the algebraic requirements for a policy to be implementable with sublinear routing tables and we give a comprehensive scalability characterization for many practically important routing policies used throughout the Internet.
- By generalizing the notion of *stretch*, we explore the algebraic conditions under which the well-known shortest-path-based constant-stretch compact routing schemes generalize to policy routing [49, 209].
- We give the first negative result showing that certain routing policies do not admit sublinear size routing tables even for arbitrary constant stretch.

The rest of this Chapter is structured as follows. In Section 4.2, we introduce the basic notations and models used throughout this Chapter. Next, in Section 4.3 we characterize the local memory requirements for implementing an important subset of routing algebras, called delimited regular algebras, and we apply the results to real-world routing policies. In Section 4.4 we deal with an algebraic interpretation of stretch and we generalize compact routing algorithms to regular algebras. Finally, in Section 4.5 we briefly highlight some further results that we obtained using the new algebraic compact routing framework and we position our work in the compact routing literature.

4.2 Formal Model

Next, we introduce a formal model for compact policy routing. Since the basic definitions will often differ from the ones used in the previous chapters (e.g., from now on we consider undirected graphs instead of directed ones), we re-define the terminology and the notation for most of the concepts relevant in this context. We also introduce the abstract algebraic framework that we shall use to cast our results.

4.2.1 Notation and Definitions

Let the communications network be modeled as a finite, connected, simple, undirected graph $G(V, E)$, let $|V| = n$ and let $|E| = m$. Communication between nodes is carried out by sending packets: neighboring nodes exchange packets directly, while remote nodes communicate through intermediate hops. We assume that nodes v (edges e) are uniquely identified by a nonnegative integer $id(v)$ ($id(e)$); we shall often write the short-hand v (e) in place of $id(v)$ ($id(e)$, respectively). Let $\delta(v)$ denote the degree of node $v \in V$ and let $d = \max_{v \in V} \delta(v)$. An $s - t$ *walk* is a sequence of nodes $p = (s = v_1, v_2, \ldots, v_k = t)$, where k is the length of the walk and $(v_i, v_{i+1}) \in E : \forall i = 1, \ldots, k-1$. A *cycle* is a walk with $s = t$, and a *path* is a walk that visits a node at most once. See Table 4.1 for a summary on the notations used in this Chapter.

Policy routing. Generally speaking, a routing policy can be considered as a function $p^*_{st} = \mathrm{Pol}(\mathcal{P}_{st})$ that from the set of available $s - t$ paths $\mathcal{P}_{st}$ selects a single *preferred* path p^*_{st} according to some predefined rules. This definition is broad enough to contain basically every conceivable policy, including extreme cases like choosing a random path as well as traditional ones like shortest-path routing.

Table 4.1: Notation.

$G(V, E)$	a simple connected undirected graph, with the set of nodes V ($\|V\| = n$) and the set of edges E ($\|E\| = m$)
$id(v)$, $id(e)$	identifier of node $v \in V$ or edge $e \in E$
$\delta(v)$	the degree of node $v \in V$
$p = (s = v_1, v_2, \ldots, v_k = t)$	an $s - t$ walk, cycle, or path of length k
$\mathcal{P}_{st}$	the set of all $s - t$ paths available in G
$p^*_{st} = \mathrm{Pol}(\mathcal{P}_{st})$	a routing policy, a function that maps from $\mathcal{P}_{st}$ to a single preferred path p^*_{st}
p^*_{st}	the preferred $s - t$ path by a policy Pol

Routing algebras. Below, we leverage the abstract notion of *routing algebras* from Sobrinho and Griffin to describe routing policies [39, 83, 86, 91, 191, 192]. This allows us to infer generic properties instead of having to define particular routing policies one by one and building piecemeal compact routing frameworks for each one separately. In addition, it has been shown that basically all practically important routing policies possess an algebraic representation [86]. Thus, we shall use the terms routing policy and routing algebra interchangeably below.

A routing algebra abstracts away the most important concepts of shortest-path routing, namely *weight composition*, the method of constructing the weight of a path from the weights of its constituent edges, and *weight comparison*, expressing the preference between edges or paths. Formally, the following properties are presumed.

Definition 4.1. A routing algebra $\mathcal{A}$ is defined as a totally ordered commutative semigroup with a compatible infinity element:

$$\mathcal{A} = (W, \phi, \oplus, \preceq) \ ,$$

where W is the set of (abstract) weights that can be assigned to edges, $\phi \in W$ is a special infinity weight meaning that an edge/path is not traversable, $\oplus$ is a composition operator for weights, and $\preceq$ is weight comparison.

In addition, $\mathcal{A}$ satisfies the below requirements.

- $(W, \oplus)$ is a commutative semigroup
  - Closure: $w_1 \oplus w_2 \in W$ for all $w_1, w_2 \in W$
  - Associativity: $(w_1 \oplus w_2) \oplus w_3 = w_1 \oplus (w_2 \oplus w_3)$ for all $w_1, w_2, w_3 \in W$
  - Commutativity: $w_1 \oplus w_2 = w_2 \oplus w_1$ for all $w_1, w_2 \in W$
- $\preceq$ is a total order on W
  - Reflexivity: $w \preceq w$ for any $w \in W$
  - Anti-symmetry: if $w_1 \preceq w_2$ and $w_2 \preceq w_1$, then $w_2 = w_1$ for any $w_1, w_2 \in W$
  - Transitivity: if $w_1 \preceq w_2$ and $w_2 \preceq w_3$, then $w_1 \preceq w_3$ for any $w_1, w_2, w_3 \in W$
  - Totality: for all $w_1, w_2 \in W$ either $w_1 \preceq w_2$ or $w_2 \preceq w_1$
- ϕ is compatible with $(W, \oplus)$ according to $\preceq$
  - Absorptivity: $w \oplus \phi = \phi \oplus w = \phi$ for all $w \in W$
  - Maximality: $w \prec \phi$ for all $w \in W \setminus \{\phi\}$

In our framework, routing policies are represented by assigning abstract weights to edges and defining path preference through the relation $\preceq$ on these weights. Given a path $p = (v_1, v_2, \ldots, v_k)$ we obtain the weight $w(p)$ of p by combining the weight of its constituent edges:

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

Then, a *preferred* path in the algebra $\mathcal{A}$ between two nodes s and t is simply the one with the smallest weight according to the relation $\preceq$:

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

We assume trat ties are broken arbitrarily but deterministically, so that all traffic demand for an $s-t$ pair is satisfied over a unique unsplittable path.

One easily checks that shortest-path routing corresponds to the algebra $(\mathbb{N}, \infty, +, \leq)$, while widest-path routing, where preferred paths are those with the largest bottleneck capacity, is simply $(\mathbb{N}, 0, \min, \geq)$. See further examples later in Section 4.3.2.

Regularity. A special family of routing algebras, called *regular* routing algebras, will play an essential role below.

Definition 4.2. A routing algebra $\mathcal{A} = (W, \phi, \oplus, \preceq)$ is said to be *regular* if it satisfies the following properties.

- Monotonicity (M): $w_1 \preceq w_2 \oplus w_1$ for all $w_1, w_2 \in W$.
- Isotonicity (I): $w_1 \preceq w_2 \Rightarrow w_3 \oplus w_1 \preceq w_3 \oplus w_2$ for all $w_1, w_2, w_3 \in W$.

Note that in what follows we adopt the terminology and definitions of Sobrinho [191], with the understanding that different authors may adopt different terminology. For instance, what will be called *isotonicity* here is called *monotonicity* in conventional order theory. The reason is that Sobrinho's terminology seems to be more broadly used in the literature.

Monotonicity (M) means that prepending an edge (or path) of weight w_1 with another edge (or path) of w_2 can only make it less preferred: $w_2 \oplus w_1 \succeq w_1$. By commutativity, the same applies to appending edges/paths: $w_1 \oplus w_2 \succeq w_1$. Isotonicity (I), on the other hand, requires $\preceq$ to be compatible with the semigroup $(W, \oplus)$ in the following sense: if an edge/path is preferred over some other one, then prepending or suffixing both with a common edge or path maintains this relation. Isotonicity, consequently, formulates in pure algebraic terms the interesting property that any subpath of a preferred path is also preferred.

Below are some further algebraic properties we shall often use to characterize routing policies [91].

Definition 4.3. Consider the following properties of a routing algebra $\mathcal{A} = (W, \phi, \oplus, \preceq)$.

- Delimited (D): $w_1 \oplus w_2 \neq \phi$ for all $w_1, w_2 \in W \setminus \{\phi\}$.
- Strictly monotone (SM): $w_1 \prec w_2 \oplus w_1$ for all $w_1, w_2 \in W$
- Selective (S): $w_1 \oplus w_2 \in \{w_1, w_2\}$ for each $w_1, w_2 \in W$.
- Cancellative (N): $w_1 \oplus w_2 = w_1 \oplus w_3 \Rightarrow w_2 = w_3$ for each $w_1, w_2, w_3 \in W$.
- Condensed (C): $w_1 \oplus w_2 = w_1 \oplus w_3$ for each $w_1, w_2, w_3 \in W$.

From the above, perhaps only delimitedness deserves more explanation. This property ensures that edges can be combined in an arbitrary sequence without the risk of obtaining an untraversable path. Intra-domain routing policies, like shortest-path routing or widest-path routing, are usually delimited, while inter-domain routing policies, like the ones used in the Border Gateway Protocol (BGP), are usually not.

Composing and decomposing routing algebras. An attractive feature of routing algebras is that surprisingly complex and expressive policy constructions can be built using only an elemental set of primitive algebras, by applying simple algebra composition and decomposition operators [86]. Two of these operators have particular importance in our context, namely lexicographic products [91] and subalgebras.

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

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

Fixing a unique absorptive element for the lexicographic product algebra may pose a challenge [91]. In the below we concentrate on delimited algebras and we assume that the input graph instances are such that all link weights are finite; thus, we can set $\phi = (\phi_\mathcal{A}, \phi_\mathcal{B})$ as the infinity weight of the lexicographic product algebra without loss of generality.

As a simple example, consider the so called widest-shortest-path policy, defined as $(\mathbb{R}^+, \infty, +, \leq) \times (\mathbb{R}^+, 0, \min, \geq)$, i.e., the lexicographic product of the shortest-path and the widest-path routing algebras [6]. Here, edge costs and edge capacities are composed separately and path preference is decided by edge costs with tie-breaking between equal cost shortest-paths on the path capacity. Note that order matters: the lexicographic product of the shortest-path and widest-path routing algebras taken in the reverse order, that is, $(\mathbb{R}^+, 0, \min, \geq) \times (\mathbb{R}^+, \infty, +, \leq)$ yields another routing policy called shortest-widest-path routing [140, 219], with completely different scaling properties (see later).

Proposition 4.4. The lexicographic product operator transforms the properties of the constituent algebras according to the following rules [91]:

- $\mathrm{M}(\mathcal{A} \times \mathcal{B}) \Leftrightarrow \mathrm{SM}(\mathcal{A}) \vee (\mathrm{M}(\mathcal{A}) \wedge \mathrm{M}(\mathcal{B}))$
- $\mathrm{I}(\mathcal{A} \times \mathcal{B}) \Leftrightarrow \mathrm{I}(\mathcal{A}) \wedge \mathrm{I}(\mathcal{B}) \wedge (\mathrm{N}(\mathcal{A}) \vee \mathrm{C}(\mathcal{B}))$
- $\mathrm{SM}(\mathcal{A} \times \mathcal{B}) \Leftrightarrow \mathrm{SM}(\mathcal{A}) \vee (\mathrm{M}(\mathcal{A}) \wedge \mathrm{SM}(\mathcal{B}))$

The second algebra composition operator we consider is subalgebras. Given a routing algebra $\mathcal{A} = (W, \phi, \oplus, \preceq)$ and a weight set $W' \subseteq W$, the restriction of $\mathcal{A}$ to W': $(W', \phi, \oplus, \preceq)$ is a subalgebra of $\mathcal{A}$ if and only if W' is closed for $\oplus$. Subalgeras inherit the properties of the root algebra, but new ones may also emerge. For instance, the subalgebra $(\mathbb{Z}^+, \infty, +, \leq)$ of the weakly monotone algebra $(\mathbb{Z}^+ \cup \{0\}, \infty, +, \leq)$ is strictly monotone.

Routing model. In order to describe the complex process of policy routing and forwarding, we generalize the model of *routing functions* from [73, 80]. Note that routing functions in compact routing connote a completely different concept than routing functions in multipath rate-control as used in the previous Chapter; unfortunately, the divergent terminologies across different disciplines frequently produce such unfortunate name

collisions in interdisciplinary research. In this model, a packet contains a payload plus a header[5] with routing related information. Now, given a routing policy $\mathcal{A}$ and a graph G, a *policy routing function* is a mapping $R : \mathbb{N} \times \mathbb{N} \mapsto \mathbb{N} \times \mathbb{N}$ together with a labeling of the nodes $L_V : V \mapsto \mathbb{N}$ and a labeling of the edges $L_E : E \mapsto \mathbb{N}$ with the following property: for each node pair s and t for which a traversable $s - t$ path exists (i.e., a path whose weight is not equal to ϕ), the successive application of R

$$(h_{i+1}, l_{i+1}) = R(v_i, h_i), \ \ \forall i = 1, \ldots, k-1$$

yields a preferred path $p^*_{st} = (s = v_1, \ldots, v_i, \ldots, v_k = t)$ according to $\mathcal{A}$ and corresponding edge labels $l_{i+1} = (v_i, v_{i+1})$, where h_1 is some appropriate initial header. We shall say that R *implements* $\mathcal{A}$ on G for indicating that R produces preferred paths according to $\mathcal{A}$ on G.

Similarly to [73, 80], we assume that node labels (or addresses) can be encoded on $c \log n$ bits[6] for some c constant. We further assume that for each node $v_i \in V$ the edges emanating from v_i are labeled locally: $L_E(v_i, v_j) \in \{1, \ldots, \delta(v_i)\}$. Additionally, the edge label l_{i+1} is understood as coming from the local label space $L_E(v_i)$ of v_i. These limitations are to ensure that no extra routing information can be encoded in the labels besides pure identification. No such limitation exists, however, on the header size.

Routing according to the policy routing function R occurs as follows. Upon receiving a packet with header h, a node u simply evaluates its *local routing function* $R_u(h) = R(u, h)$ to obtain a new header h' and an outgoing port at edge l. Then, u sets the packet's header to h' and forwards it on l. In general, this routing model is suitable to represent oblivious routing architectures, i.e., ones in which the route of a packet depends only on the contents of the packet itself and some static forwarding information. Yet, it is broad enough to capture basically any practically relevant forwarding scheme, like traditional destination-based and source-destination-based forwarding, label swapping, etc. For further details, consult [73, 80].

Memory requirements of implementing a routing policy. Introducing routing functions makes it possible to comfortably characterize the local memory needed at network nodes to implement a routing policy.

Definition 4.5. The local memory requirement $M_{\mathcal{A}}$ of implementing the routing policy $\mathcal{A}$ is defined as:

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

where $M_{\mathcal{A}}(R, u)$ is the minimum number of bits needed to encode the local routing function R_u, $\mathcal{R}$ is the set of all policy routing functions implementing $\mathcal{A}$ on some graph G, and $\mathcal{G}_n$ is the set of all graphs of size n.

A routing policy is said to be *incompressible*, if $M_{\mathcal{A}}$ is $\Omega(n)$. Otherwise $\mathcal{A}$ is *compressible*. Easily, an incompressible routing policy does not scale well, as the memory needed to store the local routing process of some node increases with the number of nodes in at least one graph. On the other hand, compressible routing policies scale well.

[5] Without loss of generality, headers can be represented by natural numbers.
[6] Logarithms are of base 2.

4.2.2 Problem Formulation

At this point, we have all the definitions in place to focus on our main concern what we call *algebraic compact routing*: given a routing algebra describing a particular routing policy, *(i)* identify the theoretical bounds on the memory requirements needed to implement that algebra, and *(ii)* examine the local storage vs. path optimality trade-off. This trade-off involves designing compact routing schemes that implement the algebra with sublinear local storage at the price of letting traffic to be routed along non-preferred paths, whose suboptimality is upper bounded by a suitably defined "abstract" stretch.

From the standpoint of routing, regular algebras manifest the "well-behaved cases" [39, 191, 192]. Monotonicity and isotonicity, on the one hand, guarantee that the preferred paths themselves can be obtained in polynomial time using a generalization of Dijkstra's algorithm. On the other hand, in a regular algebra preferred paths terminating at a given node make up a tree, allowing for a single routing entry to be maintained with respect to each node and forwarding packets based on the destination address only. This allows us to store local routing information on at most $\tilde{\mathrm{O}}(n)$ bits local memory. We formulate these ideas as follows.

For some graph G and algebra $\mathcal{A}$, define a *destination-based routing function* $\hat{R}$ for implementing $\mathcal{A}$ on G as follows. Let the packet header consist of the identifier of the packet's destination and let node u forward a packet destined to some v on the first edge l_v along the preferred path p^*_{uv}: $\hat{R}_u(v) = (v, l_v)$. Sobrinho makes the following observation [192]:

Proposition 4.6. $\mathcal{A}$ can be implemented by a destination-based routing function on any graph, if and only if $\mathcal{A}$ is regular.

One easily sees that $\hat{R}$ basically corresponds to destination oriented routing tables, storing a single entry for each destination node. This leads to the following observation.

Observation 4.7. If $\mathcal{A}$ is regular, then it can be implemented using $O(n \log d)$ bits local information.

A key question in compact routing research is whether this trivial routing function is optimal in the sense that it requires the minimum possible local memory to encode preferred paths, or there are better algorithms using less local space. For shortest-path routing in particular, Fraigniaud and Gavoille present the following negative result [73,80].

Proposition 4.8. The shortest-path algebra $\mathcal{S} = (\mathbb{Z}^+, \infty, +, \leq)$ is incompressible.

For shortest-path routing at least, routing tables are optimal. For general routing policies beyond shortest-path routing, no such characterization exists. Therefore, in the rest of this Chapter *we provide a comprehensive algebraic characterization for the memory requirements of general policy routing.* Below we discuss the case of intra-domain routing policies and delimited routing policies; see [88, 175, 176, 236] for a detailed exposition on our results for non-delimited algebras that arise in inter-domain routing, e.g., BGP policy routing.

4.3 Scalability of Delimited Routing Policies

In what follows, we discuss the algebraic requirements for a routing policy to be implementable with sublinear local storage and we also give negative results indicating

incompressibility of some practically important routing policies. As indicated above, we concentrate on delimited algebras exclusively; recall, this property ensures that finite weights combine to finite weights, implying that any concatenation of traversable paths is guaranteed to yield a traversable path.

4.3.1 Compressibility Characterizations

First, we discuss an important family of delimited routing algebras: monotone and selective algebras.[7]

Theorem 4.9. If $\mathcal{A}$ is selective and monotone then it is compressible.

In fact, we shall prove a bit more. We shall show that if a routing policy is selective, then a "preferred" spanning tree always exists with the property that for any $s, t \in V$ the only path p_{st} contained in the tree is a preferred path. We say that algebra $\mathcal{A}$ *maps to a tree*, if for any connected graph and any weighing of the edges one can always find such a "preferred" spanning tree. Then, compressibility follows as routing over a tree is possible with $\log n$ bits local memory [74].

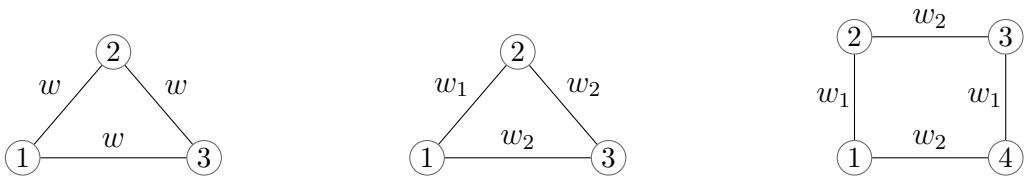
Lemma 4.10. If $\mathcal{A}$ is monotone and selective, then $\mathcal{A}$ maps to a tree. On the other hand, if $\mathcal{A}$ is delimited and $\mathcal{A}$ maps to a tree, then $\mathcal{A}$ is monotone and selective.

Proof. First, we show that if an algebra $\mathcal{A}$ is monotone and selective then it maps to a tree. Under these assumptions on $\mathcal{A}$, we construct an optimal spanning tree containing only preferred paths over $\mathcal{A}$. Start with an empty tree T, take the edges in the non-decreasing order of weights according to $\preceq$, add an edge to the spanning tree T if no cycle arises, and terminate when T spans G. We show that the only in-tree path p_{st}^T between any two nodes s and t is a preferred path over $\mathcal{A}$. To see this, take any other $s-t$ path p_{st} in G. Due to the way the algorithm proceeds, there is at least one edge (u, v) in p_{st} so that $w(u, v) \succeq w(i, j)$ for all (i, j) in p_{st}^T. Then, due to selectivity $w(p_{st}^T) \in \{w(i, j) : (i, j) \text{ in } p_{st}^T\}$, and by monotonicity $w(p_{st}^T) \preceq w(u, v) \preceq w(p_{st})$, therefore p_{st}^T is a preferred $s-t$ path. This proves sufficiency.

We prove the second statement by contraposition. In particular, we show that if a delimited algebra $\mathcal{A}$ is either non-monotone or non-selective, then in some graphs preferred paths do not reside in a tree. Obviously, if $\mathcal{A}$ is not monotone then the preferred paths might contain loops. If, on the other hand, $\mathcal{A}$ is monotone but not selective, then $\mathcal{A}$ either contains a weight $w \in W$ so that $w \oplus w \succ w$ (auto-selectivity), or $\mathcal{A}$ contains two weights $w_1, w_2 \in W, w_1 \preceq w_2$, so that $w_1 \oplus w_2 \succ w_2$. We distinguish the following cases:

- $w \oplus w \succ w$: for the case when $\mathcal{A}$ violates auto-selectivity, Fig. 4.1a gives a graph in which the preferred paths are exactly the direct edges and hence do not make up a tree;
- $w_1 \prec w_2$ and $w_1 \oplus w_2 \succ w_2$: Fig. 4.1b gives a graph where again preferred paths are via the direct edges and so no optimal tree arises;
- $w_1 = w_2$ and $w_1 \oplus w_2 \succ w_2$: in the graph of Fig. 4.1c, preferred paths are again precisely the direct edges. To see this, we only need to see that *(i)* $w_1 \prec w_2 \oplus w_1 \oplus w_2$, but this follows from $w_2 \oplus w_1 \oplus w_2 \overset{\text{(M)}}{\succeq} w_1 \oplus w_2 \succ w_2 = w_1$; and *(ii)* $w_2 \prec w_1 \oplus w_2 \oplus w_1$ can be seen similarly. Note that for the source-destination pairs that do not reach

[7] Note that selectivity implies delimitedness.



(a) $w \oplus w \succ w$ (b) $w_1 \prec w_2$, $w_1 \oplus w_2 \succ w_2$ (c) $w_1 = w_2$, $w_1 \oplus w_2 \succ w_2$

Figure 4.1: Counter-examples for different violations of selectivity.

each other via a direct edge any two-hop path is a traversable preferred path, as $w_1 \oplus w_2 = w_2 \oplus w_1 \prec \phi$ due to delimitedness. □

Note that delimitedness in Lemma 4.10 is important, as one easily finds non-delimited algebras that map to a tree even without being selective, under the assumption that each node has a finite-weight path to each other node. Further note that a special case of this result for minimum- and maximum-type of weight composition operators appeared in [12], and [83] gives similar results for special routing algebras called *dioids*.

Theorem 4.9 suggests that routing policies characterized by selective algebras can be implemented using tree routing schemes [74, 209], needing only logarithmic sized local storage (see concrete examples in the next section). In contrast to selective algebras however, there exists an important family of routing policies that, similarly to shortest-path routing, can only be implemented using at least $\Omega(n)$ bits local memory.

Theorem 4.11. If $\mathcal{A}$ is delimited and strictly monotone then it is incompressible.

We shall prove a more general claim, of which the above is a simple corollary.

Lemma 4.12. If $\mathcal{A}$ contains a delimited strictly monotone subalgebra, then $\mathcal{A}$ is incompressible.

Proof. We trace back incompressibility to the incompressibility of minimum-hop routing (Proposition 4.8), by showing that a delimited, strictly monotone algebra has subalgebras with the same algebraic structure as shortest-path routing. We use the following basic facts from semigroup theory [45]. Every element $w \in W$ of a semigroup $(W, \oplus)$ generates a subsemigroup, the so called cyclic semigroup, $(W_w, \oplus) : W_w = \{w, w^2, w^3, \ldots\}$ through the power operation:

$$\forall n \in \mathbb{Z}^+ : \quad w^n = \begin{cases} w & \text{if } n = 1 \\ w \oplus w^{n-1} & \text{otherwise} \end{cases}$$

If the ordered semigroup $(W, \oplus, \preceq)$ is delimited and strictly monotone, then any of its cyclic subsemigroups $(W_w, \oplus)$ is of infinite order, in which case it is isomorphic to the semigroup $(\mathbb{Z}^+, +)$ of positive integers under addition through the mapping $f : \mathbb{Z}^+ \leftrightarrow W_w$, $f(n) = w^n$. In addition, f is also an order preserving isomorphism between the shortest-path routing algebra $S = (\mathbb{Z}^+, \infty, +, \leq)$ and $(W_w, \phi, \oplus, \preceq)$ in this case, as $i < j \Leftrightarrow w^i \prec w^j$ due to strict monotonicity. One easily checks this by observing that for any $i < j : w^i \prec w^i \oplus w = w^{i+1} \preceq w^j$. Thus, if $\mathcal{A} = (W, \phi, \oplus, \preceq)$ has a strictly monotone subalgebra, then for any graph G and any labeling of the edges of G by positive integers

Table 4.2: Local memory requirements of various routing policies.

Algebra	**Definition**	**Properties**	**Local memory**
Shortest path	$\mathcal{S} = (\mathbb{Z}^+, \infty, +, \leq)$	SM, I	$\Theta(n)$
Widest-path	$\mathcal{W} = (\mathbb{R}^+, 0, \min, \geq)$	S, I, M	$\Theta(\log n)$
Most reliable path	$\mathcal{R} = ((0,1], 0, *, \geq)$	SM, I	$\Theta(n)$
Usable path	$\mathcal{U} = (\{1\}, 0, *, \geq)$	S, I, M	$\Theta(\log n)$
Widest-shortest-path	$\mathcal{WS} = \mathcal{S} \times \mathcal{W}$	SM, I	$\Theta(n)$
Most reliable widest-shortest path	$\mathcal{MRWS} = \mathcal{R} \times \mathcal{WS}$	SM, $\neg I$	$\Omega(n)$
Shortest-widest-path	$\mathcal{SW} = \mathcal{W} \times \mathcal{S}$	SM, $\neg I$	$\Omega(n)$

as weights, we can construct a labeling using weights from W so that a path is a shortest path in the algebra $\mathcal{S} = (\mathbb{Z}^+, \infty, +, \leq)$ if and only if it is a preferred path in $\mathcal{A}$. This implies that routing in $\mathcal{A}$ requires at least as much local memory as shortest-path routing (i.e., $\Omega(n)$ by Proposition 4.8), which completes the proof. □

4.3.2 Applications

We list some of the relevant intra-domain routing policies studied most extensively in the literature in Table 4.2, together with the algebraic definition, basic properties, and the local memory requirements as indicated by the above theoretical results. Note that all listed algebras are delimited, and they are also regular except the last one which is non-isotone. Here, $\mathcal{S}$ is the well-known shortest-path routing algebra, for which Proposition 4.8 provides an adequate incompressibility characterization. Easily, Theorem 4.11 gives the same characterization.

$\mathcal{W}$ denotes the widest-path routing policy [219]. Here, the weight of an edge is its capacity, the end-to-end capacity of a path equals the bandwidth of its bottleneck edge (the one with the smallest capacity) and the higher the capacity along a path the more preferred. This corresponds to the selective regular algebra $(\mathbb{R}^+, 0, \min, \geq)$, and so $\mathcal{W}$ is compressible by Theorem 4.9. In particular, under the tree routing scheme due to Fraigniaud and Gavoille [74], widest-path routing can be implemented using $5 \log n$ bit addresses and $3 \log n$ bits local memory, or $\log^2 n$ bits using the scheme due to Thorup and Zwick [209]. Similar is the case for the usable-path routing strategy ($\mathcal{U}$), applied extensively in Ethernet switching.[8] However, the rest of the routing policies listed in the table are incompressible.

Most-reliable-path routing ($\mathcal{R}$) denotes the policy when edges are assigned a reliability metric denoting the possibility that a packet will be transmitted successfully over the edge, and the path with the highest probability of success is favored. Easily, $\mathcal{R}$ contains the delimited strictly monotone subalgebra $((0,1), 0, *, \geq)$ and thereby it is incompressible by Theorem 4.11. Widest-shortest-path ($\mathcal{WS}$) routing prefers from the set of shortest-paths the one with the highest free capacity [6], and shortest-widest-path ($\mathcal{SW}$, [140,219]), just contrarily, prefers the shortest one out of the set of widest paths. These algebras can be expressed as lexicographic products of the $\mathcal{S}$ and $\mathcal{W}$ algebras and, by Proposition 4.4, strictly monotone [91]. Hence, for $\mathcal{R}$ and $\mathcal{WS}$, which are isotone, Theorem 4.11 supplies the local memory requirement of $\Omega(n)$. This characterization is tight apart from a logarithmic factor, as simple table-based destination-oriented routing requires $O(n \log n)$

[8]The fact that Ethernet runs over what is called the Spanning Tree Protocol highlights the expressiveness of Lemma 4.10.

bits by Observation 4.7. On the other hand, $\mathcal{SW}$ is not isotone. Theorem 4.12 holds for non-isotone algebras as well, which supplies a $\Omega(n)$ bits local memory requirement for $\mathcal{SW}$; nevertheless, at the moment it is an open question whether this characterization is tight as the trivial routing function for $\mathcal{SW}$ stores a separate routing table entry for each source-destination pair, which needs $O(n^2 \log n)$ bits per router.

4.4 Compact Policy Routing

As has been shown in the previous section, many practically relevant routing policies are impossible to implement with sublinear size routing tables. In the case of shortest-path routing, a standard way to improve scalability is to define *compact routing schemes*. In these schemes, paths are allowed to be longer than the shortest one, but path increase is upper bounded by a *multiplicative stretch factor* k, meaning that the paths yielded by the compact routing scheme are at most k times as long as the shortest one. In the followings, we characterize the routing policies that admit similar compact implementations, at least for a sufficient abstract notion of stretch.

4.4.1 Compact Routing on Regular Algebras

We start with an algebraic generalization of the notion of multiplicative stretch.

Definition 4.13. A routing scheme is of stretch k over algebra $\mathcal{A}$, if for any path p_{st} selected by the scheme: $w(p_{st}) \preceq (w(p^*_{st}))^k$, where p^*_{st} is some preferred $s-t$ path in $\mathcal{A}$ and

$$(w(p^*_{st}))^k = \underbrace{w(p^*_{st}) \oplus w(p^*_{st}) \cdots \oplus w(p^*_{st})}_{\text{k times}} \ . \tag{4.1}$$

By (4.1), the above definition indeed generalizes the notion of multiplicative stretch originally defined for shortest-path routing.

Next, we ask which routing algebras lend themselves to be implemented in sublinear space using a compact routing scheme that admits a finite stretch.

Theorem 4.14. If a routing algebra $\mathcal{A}$ is delimited and regular then there is a stretch-3 compact routing scheme for $\mathcal{A}$.

We show that the stretch-3 shortest-path routing scheme due to Cowen [49] readily generalizes to regular algebras. Below, we briefly reproduce that scheme. For further details, see [49] and [209].

For each $u \in V$, choose some node set $L \subseteq V$ and with each $u \in V$ associate a *landmark* l_u as the node closest (according to $\mathcal{A}$) to u in the set L. Additionally, for each $u \in V$ define a *ball* $B(u) = \{v \in V : w(p^*_{u,v}) \preceq w(p^*_{u,l_u})\}$, where $p^*_{s,t}$ refers to the preferred $s-t$ path for any s and t. Finally, let the *cluster* of u be $C(u) = \{v \in V : u \in B(v)\}$. When $\mathcal{A}$ is regular, one can use the lexicographic lightest-path algorithms in [191, 192] to obtain unique connected clusters for each u.

The compact routing scheme due to Cowen is a hop-by-hop technique. The label of node v consists of the triplet $(v, l_v, \text{port}_{l_v,v})$, where v is the identifier of the node, l_v is the identifier of its corresponding landmark, and $\text{port}_{l_v,v}$ is the local port at l_v to the first hop on the preferred path from l_v to v. The packet header is the label of the target node. The routing table at node $u \notin L$ consists of $(v, \text{port}_{u,v})$ tuples with respect to each $v \in C(u) \cup L$, where $\text{port}_{u,v}$ is again the local port label of the first edge along the preferred $u-v$ path.

Packet forwarding *inside* a cluster occurs along preferred paths using the entries in the local routing tables. To route a packet to a node v *outside* the cluster, node u first forwards the packet to v's landmark, from where it arrives to v using again a direct route. In particular, when a packet with target v arrives to a node $u \neq v$, u checks whether v is contained in its local routing table. If not, then l_v, the landmark of v, is extracted from the header. If $u = l_v$ then the appropriate port label is also extracted from the header, otherwise it is looked up in the local routing table. Forwarding terminates when $u = v$.

From Proposition 4.6, we know that if $\mathcal{A}$ is regular then standard destination-based hop-by-hop routing is correct. To show that the above scheme is also correct, the following crucial fact is enough (observed for shortest-path routing by Cowen in [49]).

Lemma 4.15. Suppose that $\mathcal{A}$ is monotone. Now, if u stores an entry in its local routing table towards some t, then the next hop v along the preferred p^*_{ut} path also stores an entry to t.

Proof. Easily, by monotonicity $p^*_{vt} \preceq p^*_{ut} \preceq p^*_{l_t,t}$ so v also stores an entry for t. □

Next, we show that the scheme is stretch-3 on $\mathcal{A}$. As forwarding inside clusters occurs along preferred paths, we only need to prove stretch-3 for indirect forwarding via landmarks.

Lemma 4.16. If $\mathcal{A}$ is regular, then for any $u, v \in V$ with $v \notin C(u) : w(p^*_{u,l_v}) \oplus w(p^*_{l_v,v}) \preceq (w(p^*_{u,v}))^3$.

Proof. (i) by assumption, $w(p^*_{l_v,v}) \preceq w(p^*_{u,v})$. *(ii)* by using the triangle inequality: $w(p^*_{u,l_v}) \preceq w(p^*_{u,v}) \oplus w(p^*_{v,l_v}) = w(p^*_{u,v}) \oplus w(p^*_{l_v,v})$ (the latter equlality comes by commutativity). Here, the triangle inequality represents the basic fact that for any triplet $u, v, w \in V$ the $u - w - v$ path of weight $w(p^*_{u,w}) \oplus w(p^*_{w,v})$ is a candidate for the preferred $u - v$ path $p^*_{u,v}$, and therefore $w(p^*_{u,v}) \preceq w(p^*_{u,w}) \oplus w(p^*_{w,v})$. *(iii)* using isotonicity, from *(i)* and *(ii)* we have $w(p^*_{u,l_v}) \preceq w(p^*_{u,v}) \oplus w(p^*_{u,v})$. Combining *(i)* and *(iii)* by isotonicity we obtain $w(p^*_{u,l_v}) \oplus w(p^*_{l_v,v}) \preceq w(p^*_{u,v}) \oplus w(p^*_{u,v}) \oplus w(p^*_{u,v})$. □

Finally, we show that the local information is indeed sublinear. Obviously, addresses can be encoded on $3 \log n$ bits. The size of the local routing table at node u is $O(|C(u)| + |L|)$. Using the landmark selection technique given by Cowen one obtains a local memory requirement of $O(n^{2/3})$ [49], which is improved by Thorup and Zwick to $\tilde{O}(n^{1/2})$ in [209].

Note that delimitedness is important to be able to apply Cowen's scheme. If the algebra is not delimited, then we might not be able to find landmarks reachable from each node in the first place. And even if we did, the notion "stretch-k" is not well-defined for non-delimited algebras as it would allow the stretched path to be of infinite weight. To understand why this is a problem, suppose that for some non-delimited algebra and for some $u - v$ pair $w(p^*_{u,v}) \prec \phi$ but $w(p^*_{u,v})^3 = \phi$. In such cases, the weight of the preferred $u - v$ path is finite, but the weight of the path through a landmark may be ϕ by our stretch-3 scheme, which would be clearly wrong as such a path is practically not traversable from u to v.

An interesting case is when the policy is the widest-path routing algebra $\mathcal{W}$. In this case, for any $n \in \mathbb{Z}^+$ and any $w \in W : w^n = w$. Hence, stretch-3 paths are exactly the preferred paths in this case. The same applies to any selective and monotone algebra. Thus, Theorem 4.14 in fact gives an alternative proof to the claim that monotone and selective algebras are compressible.

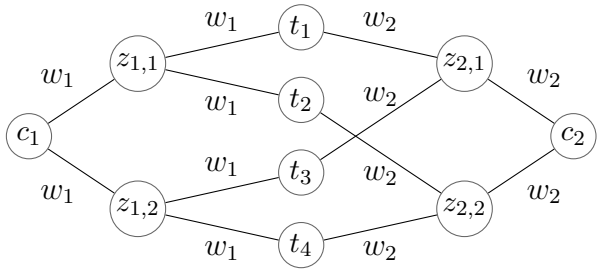


Figure 4.2: A sample graph for $p = 2$, $\delta = 2$ if the words for the target nodes are $[1, 1]$, $[1, 2]$, $[2, 1]$ and $[2, 2]$.

We argued in Section 4.2.2 that regular algebras are the "well behaved" cases from the aspect of distributed routing, as they can be implemented by destination-based routing tables. Our results so far indicate that regular algebras are "well-behaved" from the standpoint of compact routing as well: not just that we could give a general result characterizing the memory requirements for implementing regular algebras, but we also found that even when a regular algebra turns out incompressible a stretch-3 compact routing scheme is guaranteed to exist. In the next section, we show that if regularity fails then finite stretch compact routing becomes significantly more difficult.

4.4.2 Compact Routing When Isotonicity Fails

We have shown that regularity of a delimited routing algebra is sufficient to define a stretch-3 compact routing scheme. It is an intriguing question whether it is necessary as well. At the moment, we do not have an answer to this question. What we can show, however, is that when isotonicity fails in a very intricate way, then no stretch-k routing exists for any k constant.

Theorem 4.17. Let $k \geq 1$ and let $\mathcal{A} = (W, \phi, \oplus, \preceq)$ be a monotone algebra with the property that for any $p \geq 2$ there exists a set of weights $\{w_1, w_2, \ldots, w_p\} \subseteq W$ so that $\forall i, j \in \{1, \ldots, p\}, i \neq j$:

$$w_i \oplus w_j \succ w_i^{2k} \text{ and } w_i \oplus w_j \succ w_j^{2k} \ . \tag{4.2}$$

Then, there is no stretch-k routing scheme with sublinear memory requirement at all nodes.

Proof. Borrowing the idea from [73], we present a family of graphs on which any stretch-k implementation of $\mathcal{A}$ requires $\Omega(n)$ bits at some nodes. Start with a set of nodes $c_i \in C$, $|C| = p \geq 2$. To each $c_i \in C$, add $\delta \geq 2$ neighbors $z_{ij}, i \in \{1, \ldots, p\}, j \in \{1, \ldots, \delta\}$ and label the edges by w_i. Finally, add δ^p nodes $t \in T$ and connect these to the z_{ij} nodes according to the following rule: for each $t \in T$ take the alphabet consisting of the symbols $(1, \ldots, \delta)$, construct a word of length p from this alphabet and add an edge from z_{ij} to t if the ith symbol in the word is exactly j. Label any (z_{ij}, t) edge by w_i. Fig. 4.2 gives an example.

By monotonicity and (4.2), the preferred path $p^*_{c_i,t}$ from any $c_i \in C$ to any $t \in T$ is the min-hop path, so $w(p^*_{c_i,t}) = w_i \oplus w_i = w_i^2$. Fraigniaud and Gavoille in [73] show that encoding these paths in the above family of graphs requires $\Omega(n \log \delta)$ bits of storage space at the nodes in C. Intuitively speaking, the idea is that there are $2^{\Theta(n^2)}$ different graphs on n nodes in this graph family, and to encode the min-hop paths the routing algorithm

needs to be able to differentiate among them, which requires $\Theta(n)$ local storage space on at least one node. See [73] for a detailed exposition of this idea.

Unfortunately, any stretch-k compact routing scheme for k finite needs to encode the exact same min-hop paths. By construction, any non-preferred path $p_{c_i,t}$ goes through at least two edges of weight w_j for some $j \in \{1,\ldots,p\}, j \neq i$, and hence is at least of stretch k: $w(p_{c_i,t}) \succeq w_i \oplus w_i \oplus w_j \oplus w_j \overset{\text{(i)}}{=} (w_i \oplus w_j) \oplus (w_i \oplus w_j) \overset{\text{(ii)}}{\succeq} w_i \oplus w_j \overset{\text{(iii)}}{\succ} (w_i^2)^k = w(p^*_{c_i,t})^k$, where *(i)* is by associativity and commutativity, *(ii)* is by monotonicity, and *(iii)* is by (4.2). □

A key to the above result is the weight set with the special structure (4.2), an extreme form of strict monotonicity. For $k \geq 2$, (4.2) violates isotonicity, therefore the theorem does not apply to regular algebras. But to many non-regular algebras it does. For the shortest-widest-path policy in particular, one easily generates the weights w_i with the required properties. Let $w_i = (b_i, c_i)$, where b_i denotes the capacity and c_i a positive cost, and for each $i = 1,\ldots,p$ choose $b_i = i$ and let $c_i = (2k)^{i-1}$. One easily checks that this construction satisfies (4.2), since if $i < j$ then $b_i < b_j$ implies $(b_i, c_i) + (b_j, c_j) = (b_i, c_i + c_j) > (b_j, c_j)^{2k}$, while from $c_i < 2kc_i \leq c_j$ we get $(b_i, c_i + c_j) > (b_i, c_i)^{2k}$. This implies that *the shortest-widest-path policy does not admit a compact implementation for any finite stretch* by Theorem 4.17.

4.5 Related Work

Thanks to the tenacious research efforts in the field of compact routing, we now have a sufficient insight into the theoretical scalability of shortest path routing. Motivated by the fact that many routing applications adopt a significantly more complex policy to classify paths than pure shortest-path routing, we have shown an algebraic approach towards generalizing the theory of compact routing to policy routing. We presented some "landmark" theorems, which can be used as guidelines to classify routing policies based on the respective algebraic properties, and we identified some algebraic requirements for effectively trading between path preference and memory. As an important message, we identified delimitedness and regularity as the cornerstones of compact policy routing, allowing for a generic compressibility theory to be formulated as well as defining a finite stretch compact routing scheme. The fact that regular algebras are exactly the ones that can be efficiently implemented in a distributed way [86, 91, 191, 192] makes these algebras highly attractive for designing future routing policies [187].

Routing policies that are described by non-regular algebras, however, have turned out very difficult to characterize, both in terms of the memory requirements for implementing them at the network nodes as well as in terms of constant-stretch compact routing approximations. Crucially, several highly important examples of non-regular policies occur in the context of the Border Gateway Protocol (BGP), the inter-domain routing mechanism that glues the Internet together [102, 216].

In a sequel to the work presented above, we extended the framework of algebraic compact routing to the non-regular BGP policies and we provided a comprehensive characterization [175, 176]. For this, we modeled BGP policies at multiple increasingly richer levels of policy expressiveness. At the first, elemental level, BGP policy routing corresponds to the so called *provider-customer routing policy*, where the autonomous domains that constitute the large-scale structure of the the Internet, the so called *Autonomous Systems* (AS), sell wholesale transit service to each other. It turns out that the resultant

policy, under mild regularity conditions, is compressible. At the second level of BGP policy routing, we extend the pure-transit model with admitting *peer* relationships to also exist between ASes, in which nodes voluntarily exchange traffic with each other in a settlement-free manner. We show that the resultant *valley-free* routing policy is incompressible. We also find that all extensions of this policy to furtuer complexities of BGP policy routing are likewise incompressible, including *valley-free with local preference* and *valley-free with local preference and tie-breaking on AS-path length*.

Our results paint a pessimistic picture on the long-term sustainability of Internet routing, providing the first ever *theoretical evidence* that BGP policy routing, the prevalent routing policy across the Internet, is fundamentally unscalable. Our results in this regard are completely in line with the mounting *empirical evidence* on the worrying scalability issues the current-day Internet inter-domain routing ecosystem exhibits [119, 148, 233]. See more on this matter in the next Chapter.

Motivated by the fundamental incompressibility of Internet routing, in [88] we attempted to define suitable compact routing schemes for BGP policy routing. Unfortunately, most of our results are negative: not just that BGP policy routing is incompressible but essentially no "reasonable" definition of stretch exists that would admit a compressible approximation for BGP policy routing.

Finally, in [236] we extended the algebraic compact routing framework to service function chaining [93]. Here, network functions, like intrusion detection, network address translation, or video transcoding, are realized as standalone or virtualized middleboxes scattered throughout the network, and packets must pass through these functions in specific order. In the setting of service function chaining many of the assumptions we made in our algebraic framework are violated; for instance preferred paths may contain loops (i.e., they degrade to preferred *walks*) and even computing the preferred path (the preferred-walk computation problem) may prove intractable over certain algebras. Still, we were able to give a comprehensive characterization on the computational complexity and the compressibility of many practically relevant service function chaining problems.

Chapter 5

Scalable Internet Routing: The Information-theoretic Perspective

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. In this Chapter, we take a new look at the scalability of Internet routing through an information-theoretic lens. In contrast to the, mostly negative, worst-case results of the previous Chapter, our findings below are mostly positive: we introduce a new model that reduces Internet routing tables to simple sequential strings, which then lend themselves readily to an information-theoretical analysis, and we show that these string representations, although indeed "incompressible" in the *worst-case* sense, admit dramatic space-reduction using special-purpose forwarding-table compression algorithms on *particular* inputs, like e.g., graphs structured similarly as the large-scale Internet topology. We then apply our information-theoretical methodology to the problem of compressing IP forwarding tables and we show orders of magnitude space reduction beyond what is available with state-of-the-art techniques.

5.1 Preliminaries

5.1.1 Internet Packet Forwarding

Today, computer networks are built on the distributed destination-based 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 [63, 119, 148, 233]: 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 [168].

We created an Internet data plane measurement infrastructure to understand the long-

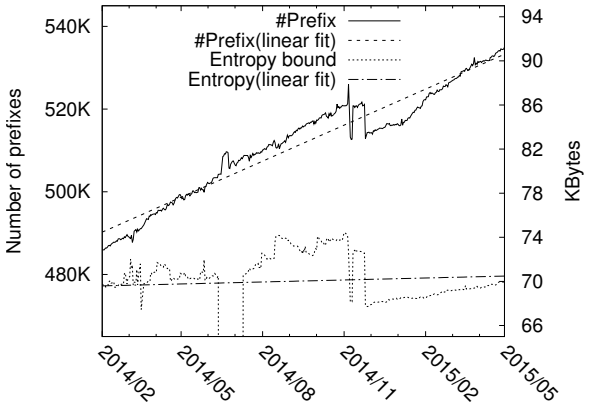


Figure 5.1: Number of IPv4 prefixes and information-theoretical entropy bound from empirical data on `rtr2.vh.hbone.hu` and the linear fit, during 18 months in 2014 and 2015.

term trends affecting Internet scalability and we publish daily statistics and downloadable datasets to the Internet community at the *Internet Routing Entropy Monitor* website.[9] We were scraping roughly two dozen IPv4 forwarding tables every day during 2014 and 2015 from operational IP routers located in the Internet Default Free Zone.[10] Throughout this time, the number of entries in the forwarding tables has grown more than 11 percent to well over half a million[11] (see Fig. 5.1). 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 we define later, 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. 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 [233].

The systematic study of the memory implications of large-scale routing was pioneered by Kleinrock and Kamoun in their seminal paper [121]. They pointed out that scalability is the central design requirement for very large networks and concluded that some form of forwarding state compression is inevitable. On the traces of McQuillan [145] they proposed a *hierarchical routing* scheme, whereby 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. In this regard, Kleinrock's work can be easily regarded as the first systematic study on compact shortest-path routing; see the previous Chapter and [127, 209]. Underlying Kleinrock's result is the observation that the assignment of node addresses may 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 [1], GSE [160], Nimrod [38], ISLAY [113]).

[9] See `http://lendulet.tmit.bme.hu/fib_comp`

[10] The DFZ is made up by the Internet core routers that maintain full BGP routing tables and can route any packet without relying on any router of "last resort", i.e., a default gateway.

[11] This figure has increased to well beyond 800,000 entries since then, as of 2020.

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 [217]. A *flat address space*, on the other hand, would introduce an extra name-to-address mapping step in all communication sessions [1]. 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. This is especially so in Internet inter-domain policy routing, where 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 stretch cannot even be defined in this setting; recall Theorem 4.17 from the previous Chapter. This seems inevitable, in that hierarchical addressing for optimal routing is provably suboptimal unless the underlying topology happens to be a tree.

It seems that we are in a trap here: our theoretical results from the previous Chapter indicate that *Internet routing is fundamentally unscalable in the worst-case sense* for all relevant routing policies, *and not even decidedly suboptimal routing helps here as long as we want to exert tight control on the allowed path-length stretch.* This situation has been recognized by the network community as well. In 2007, the "Internet Architecture Board Workshop on Routing and Addressing" [148], an invitation-only event organized by the experts responsible for the architectural oversight of Internet protocols and procedures, concluded that scalability is among the most prominent risks endangering the future growth of the Internet [148] (but see [63] for contrasting opinions). Indeed, operators are facing the consequences of rapid forwarding table growth each day, where constantly increasing fast memory on router line cards boosts silicon footprint, heat production, power budget, and the CAPEX/OPEX associated with IP network gear, and forces operators into rapid upgrade cycles [119, 233].

However, there may still be a way out of this trap: all known theoretical results so far are of *worst-case nature*, in the sense that there must be *at least one* worst-case graph where the local memory requirement of storing the forwarding table grows dramatically with the network size. But what if the Internet topology does not exhibit this worst-case behavior? What if Internet routing *in particular* is still compressible, due to some yet unknown special property of the IP address space or the high-level graph topology, despite the prohibitive worst-case characterizations? In this Chapter, we present a series of forwarding table compression schemes that prove remarkably efficient on Internet-like graphs, suggesting that maybe the Internet indeed admits substantial reduction of the routing state that has to be maintained by routers, beyond the worst-case bounds.

5.1.2 Forwarding Table Compression

Data compression is widely used in processing large volumes of information. Not just that convenient compression tools are available to curtail the memory footprint of basically any data, but these tools also come with solid information-theoretical guarantees that the compressed size is indeed minimal, in terms of some suitable notion of *entropy* [32, 48]. Correspondingly, data compression has found its use in basically all aspects of computation and networking practice, ranging from text or multimedia compression [235] to the very heart of communications protocols [222] and operating systems [27].

Traditional compression algorithms do not admit standard queries, like pattern matching or random access, right on the compressed form, which severely hinders their applicability. An evident workaround is to decompress the data prior to accessing it, but this

pretty much defeats the whole purpose. The alternative is to maintain a separate index dedicated solely to navigate the content, but the sheer size of the index can become prohibitive in many cases [100, 155].

A recent theoretical breakthrough on compressed data structures, however, offers a way to overcome these issues. A *compressed data structure* (or compressed self-index) is, loosely speaking, an entropy-sized index on some data that allows complete recovery of the original content as well as fast queries[12] on it [55,64,68,100,141,154,155,166,235]. As the compressed form occupies much smaller space than the original representation, the time required to answer a query is often far less than if the data had not been compressed in the first place [55,235]. *Compressed data structures, therefore, turn out one of the rare cases in computer science where there is no space-time trade-off.*

Compressed self-indexes, and accompanying software tools, exist for a broad set of applications; from compressors for sequential data like bitmaps (`RRR`), [166]) and text documents (`CGlimpse` [64], wavelet trees [68]); compression frontends to information retrieval systems and search engines (`MG4J` [214], `LuceneTransform` [139]) and dictionaries (`MG` [221], `Succint` [2]); to specialized tools for structured data, like XML/HTML/DOM (`XGRIND` [210], `XBZIPINDEX` [67], `Xpress` [149]), graphs (`WebGraph` [220]), 3D models (`Edgebreaker` [182]), genomes and protein sequences (`COMRi` [202]), multimedia, source and binary program code, formal grammars, etc. [221]. With the advent of replacements for the standard file compression tools (`LZgrep` [156]) and generic libraries (`libcds` [154]), compressed data structures have become mainstream.

Curiously though, compressed data structures have found limited use in networking, despite that Internet routing is affected critically by skyrocketing volumes of information to store (as we described previously). Indeed, there has been a flurry of activity to find space-efficient forwarding table representations [15, 40, 56, 59–61, 90, 95, 97, 104, 138, 143, 159, 190, 194, 197, 211, 215, 223, 231], yet very few of these go beyond ad-hoc schemes and compress to information-theoretic limits, let alone come with a convenient notion of entropy. But even these heuristic schemes have brought about an impressive reduction in forwarding table size in the recent years: from the initial 24 bytes/entry (prefix trees [190]), use of hash-based schemes [15,215], path- and level-compressed multibit tries [40,159,197], tree-bitmaps [61], etc., have reduced the memory tax to just about 2–4.5 bytes/entry [56, 211, 231]. Meanwhile, lookup performance has also improved [159].

The evident questions whether there is an ultimate limit on FIB aggregation, and whether FIBs can be reduced to fit into fast router memory entirely, have been asked several times before [40, 56, 60, 197]. But to answer these questions, we need to go beyond conventional approaches to forwarding table compression and find *(i) appropriate notions for forwarding table entropy that set clear and provable limits on forwarding table compression for particular inputs and (ii) new compressed data structures that encode to entropy-bounded space and support lookup and update in optimal time*. This Chapter is dedicated to the theory and practice of such forwarding table compression schemes.

5.1.3 Contributions

The main goal of this Chapter is to augment the worst-case characterizations from the previous Chapter and *present a systematic study of the practical memory requirements of hop-by-hop destination-based routing*. We take a principled approach: we introduce

[12]Think of a version of the venerable `gzip(1)` tool that would allow to `grep(1)` into the compressed file without explicitly deflating it first.

Table 5.1: Notation.

$G(V, E)$	a simple connected undirected graph on n nodes
δ_v	node degree of node $v \in V$
$\Sigma = \{c_1, c_2, \ldots, c_\delta\}$, $\lvert\Sigma\rvert = \delta$	alphabet of size δ
$id(v)$	a globally unique integer id for node v, $id(v) \in [1, n]$
$port(v, u)$	a locally unique (to node v) port id for edge (v, u), $port(v, u) \in [1, \delta_v]$, $(v, u) \in E$
W	address width (usually $\log n$)
T	a forwarding table in the form of a binary prefix tree
t	the number of nodes in T
L	the set of leaves in T
l	the label map $L \mapsto \Sigma = [1, \delta]$, specifying for each $v \in L$ the corresponding next-hop label $l(v) \in \Sigma$

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. This way, ours is the first study that *translates problems related to routing scalability to the language of information theory and gives verifiable space-time characterizations on forwarding table complexity for particular inputs.*

Our contributions are as follows:

- First, we concentrate on the simple model familiar from the previous Chapter where node addresses constitute an unstructured (flat) address space. In this model, forwarding tables are modeled as sequential strings, admitting tight memory requirement characterizations using Shannon's entropy measures and standard data compression techniques. We consider increasingly more complex models for address space design and we give the corresponding entropy measures as well as algorithms that attain these entropy bounds.
- Second, we consider a hierarchical routing model that is tailored to describe the Internet routing architecture. This model is much more complex than the previous one, as now the address space has an intricate internal structure and our information-theoretic framework must adequately reproduce this structure. Crucially, we can still follow our principled approach and give a descriptive and theoretically sound entropy notion to characterize the memory requirements of forwarding tables over a structured address space, and we also show the corresponding compressed data structures that provably admit the entropy bounds.

The rest of this Chapter is organized as follows. In Section 5.2 we give a short background on information theory and data compression. Then, in Section 5.3 we present our forwarding tables over flat address spaces, in Section 5.4 we extend the analysis to hierarchical address spaces, and in Section 5.5 we show analytically that the two entropy notions under these seemingly disparate models are fundamentally related to each other. Finally, Section 5.6 surveys related literature and positions are our work in the larger context of Internet scalability research.

5.2 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 [32, 48, 96].

There are various approaches to represent the empirical entropy for an input text, relative to a model of the source that generates it. In the following discussions we shall assume that this model is *static*, that is, the compression scheme can use only information that is available from the source *a priori* but 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.

5.2.1 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 (see Table 5.1 for a summary of notation). Then, encoding s is equivalent to being able to distinguish any two of the possible δ^n strings we can get as input, which needs at least

$$I(s) = \log(\delta^n) = n \log \delta \text{ bits} \tag{5.1}$$

of storage space. All logarithms below 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. The quantity $I(s)$ given by (5.1) is called the *information-theoretical lower bound* for storing s, referring to that we cannot hope for any compression beyond $I(s)$ unless the source discloses some further knowledge on the strings it generates.

5.2.2 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. Yet, we can use this limited amount of additional knowledge to compress s beyond the information-theoretical lower bound. The lower bound on the amount of memory needed to represent one symbol of s in this context is given by the *zero-order empirical entropy* of s, defined as

$$H_0(s) = \sum_{c \in \Sigma} \frac{n_c}{n} \log \frac{n}{n_c} \text{ bits} \ . \tag{5.2}$$

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

Recall, $H_0(s)$ gives the information content for a single bit of the input. For storing the *entire* input string s, the *zero-order entropy bound* of $nH_0(s)$ bits gives a trivial lower bound. Crucially, this is also an upper limit on the storage size of s as there are well-known compression schemes (e.g., Huffman coding, arithmetic coding) that attain roughly the the zero-order entropy bound; the real bound is $nH_0(s) + o(n)$ bits [32, 48].

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.[13] Consequently, *the zero-order storage bound does not depend on the actual order in which symbols appear* in s (c.f., (5.2)); e.g., using a zero-order compressor

[13] For instance, the zero-order entropy of English text is around 5 bits per character [32].

the string `mississippi` and the anagram `miiiissspp` encode to the same size, roughly 20 bits ($H_0 \sim 1.823$ in both cases), despite that the latter seems much more organized.[14]

Here, $p_c = \frac{n_c}{n}, c \in \Sigma$ is called the empirical probability distribution for s. When the succession of symbols in s is essentially random (i.e., subsequent symbols can be regarded as if chosen independently) and n is large then p_c approaches the probability of c appearing in s and the empirical entropy $H_0(s)$ converges to the conventional Shannon entropy $\sum_c p_c \log 1/p_c$.

5.2.3 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" [32, 54]. 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 $s[i-k] \dots s[i-1]$ 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*

$$H_k(s) = \sum_{c \in \Sigma} \frac{n_c}{n} \sum_{q \in \Sigma^k} \frac{n_{qc}}{n_c} \log \frac{n_c}{n_{qc}} \tag{5.3}$$

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 [69,142]. $H_k \leq H_{k-1}$ 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 $k = 1$. Using this definition, the *k-order empirical entropy bound* is simply $nH_k(s)$ bits.

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` $H_1 \sim 0.8$ bits but for the visibly more regular anagram `miiiissspp` we get only $H_1 \sim 0.6$ bits. This suggests that in a setting where we are to a certain extent free to choose the arrangement of the symbols in some string we should strive for more order, which would then directly translate into better compressibility through the notion of higher-order entropy.[15] At the extreme, if we can reorder a string into a few runs of identical symbols (like in the example `miiiissspp`) we realize maximum compression by simple run-length encoding [32, 48].

5.2.4 Compressed Data Structures

In the context of forwarding table compression, our aim is not only to squeeze as much data into as small memory as possible, 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 [180]. Traditional data compression schemes,

[14] And the latter is also much less meaningful at the same time, which nicely demonstrates how the above notion of "information content" is a purely artificial metric that has nothing to do with our anthropocentric perception of information.

[15] See e.g., the Burrows–Wheeler Transform for a practical compression scheme that rests on this observation [142]

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 [65] attain $nH_0(s) + o(n)$ bits of space and $O(\log n)$ random access, while the schemes in [69] and [16] 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 [141] 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, generic storage bounds, and higher-order string indexers, see [66, 100, 155, 235].

5.3 Forwarding Table Compression Over a Flat Address Space

In this Section, we consider a model where there is no structure in the address space, the case of *flat* addresses. In this model the fact that two addresses are close to each other in the address space (e.g., addresses are integers and the difference of two addresses is small) does not *necessarily* connote any special meaning apart from the fact that the addresses happen to be similar. This is in contrast to *hierarchical routing*, e.g., the Internet, where two nodes' proximity in the address space usually implies their proximity in the routing hierarchy, and usually also geographic proximity (see the next Section). The flat address space model is used in MPLS label switching, Ethernet forwarding, and many clean-slate routing designs [34].

5.3.1 Formal Model

We adopt the notion of *routing functions* from the previous Chapter to characterize the local memory requirement of hop-by-hop destination-based routing over a flat address space. Below, we briefly review the most important definitions and assumptions from there; see also [80] and [179].

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 assign/control the assignment of node ids will prove essential, we shall return to this crucial question soon. 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[16] in the range $[1, \delta_v]$.

We further presume that some routing policy (e.g., shortest-path, min-hop path, valley-free) 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, apart from that at each node

[16]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 [8]. In our model these two cases coincide.

it assigns a single, well-defined output port with respect to each other node in the network, so that packets destined to that node must be forwarded via the corresponding port assigned by the policy.

The routing function R_v at node v is the usual destination-based hop-by-hop routing function (recall Section 4.4.1). Packet headers contain the id of the destination node, say, u, and the local routing function R_v at each node v stores the output port to pass the packet on whenever the destination id in the packet header is exactly u. In particular, the routing function $R_v : [1, n] \mapsto [1, \delta_v]$ maps a destination node id to the corresponding outgoing port at node v, more closely, the local port id of the corresponding link. 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 set $R_v(id(v))$ arbitrarily.

It is convenient to think of R_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. To stress this string-view, from now on we shall use the string-notation s_v to denote the local routing function of a node v. 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 5.2.2). Modifications, furthermore, are simple string updates.

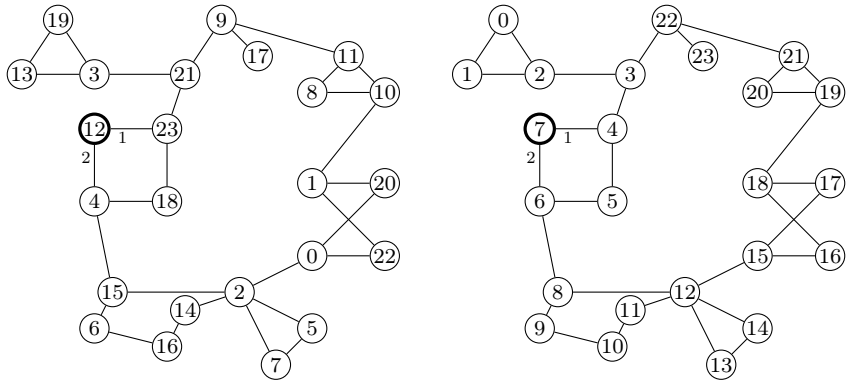
Example 5.1. A sample network and the corresponding shortest-path routing functions for two node address assignments are given in Fig. 5.2; in particular, Fig. 5.2a shows a random node id assignment and Fig. 5.2b gives a node id assignment that we chose for a particular purpose that we reveal later. Consider the first example in Fig. 5.2a and consider the thick node, with id 12. The routing function s_{12} of node 12 is also given in the figure. Now, suppose that node 12 receives a packet with a header that identifies node 3 as the destination. Then, forwarding occurs as follows: node 12 looks up its routing function s_{12} at position 4 (one plus the destination node id 3 since node ids are of base 0), finds the next-hop label 1 at this position, and hence forwards the packet at edge 1 to node 23 along the shortest path. Node 23 in turn will likewise consult its routing function s_{23} and forwards the packet at the port corresponding to the symbol at position 4 in s_{23}.

This example shows that the "source model", the abstract model that defines how node ids are generated, largely determines the attainable storage size of forwarding tables. Below, we enumerate several increasingly stronger models for network address space layout and we specify the corresponding entropy bounds for storing the routing functions in the specific address space model, along with the suitable compression schemes that attain these entropy bounds.

5.3.2 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 $\Sigma_v = [1, n]$. We are bound to assume $|\Sigma_v| = n$, since the only upper limit on the alphabet size that holds in any simple graph is that the maximum degree is at most $n - 1$, plus we need another symbol to distinguish packets destined to the node itself.

It is then easy to see that the graph-independent case maps to the no-information model of Section 5.2.1, and the respective storage size characterizations bring over as follows.



$s_{12} = 22212222111101222112121$ $s_7 = 11111220222222222211111$

(a) random address space (b) optimized address space

Figure 5.2: Sample graph from Kleinrock's original paper [121] with port ids and routing function for the thick node over different address spaces; (a) in the name-independent model with randomly assigned flat node ids (average zero-order routing function entropy $\bar{H}_1 \sim \bar{H}_0 = 1.1$ bit); and (b) in the name-dependent model with "optimized" flat node ids minimizing the first-order entropy by brute force search ($\bar{H}_0 \gg \bar{H}_1 = 0.355$ bit). Observe that nearby nodes are mapped to close-to-each-other node ids in the optimized address space (b), which translates into three-fold space reduction compared to the random address space case (a).

Theorem 5.1. Given a graph G on n nodes, for any $v \in V$ storing the routing function s_v requires at least $n \log n$ bits. In addition, there is an encoding scheme that stores s_v in at most $n \log n$ bits and supports lookups in $O(1)$ time.

Proof. The proof is trivial. By (5.1), $n \log n$ bits of information is needed to at least differentiate between any two string instances when the alphabet size is n, so $n \log n$ is a lower bound in the graph-independent model. At the same time this is also an upper bound: any naive sequential string encoding will attain the information-theoretical lower bound that stores each next-hop symbol on $\log n$ bits, yielding $n \log n$ bits space for the entire routing function and supporting $O(1)$ access to a random position. □

As we have shown in the previous Chapter, this bound coincides with the general worst-case *lower bound* for shortest-path routing as per Gavoille and Pérennès (see Proposition 4.8), which we proved to be valid for many routing policies beyond shortest paths (recall Theorem 4.11 and Table 4.2). As we argued therein this is bad news, as super-linearly scaling memory for mere packet forwarding in a network growing as rapidly as the Internet would put routers under endless memory stress [233]. Things are not that hopeless though because *particular graphs and routing policies* may admit significant space-reduction beyond the prohibitive worst-case, as we show next.

5.3.3 Name-Independent Case

Consider the following model. Suppose we know the input graph G of size n, the next-hop alphabet $\Sigma_v = [1, \delta_v]$ for each $v \in V$, and the routing policy that assigns the paths. However, we assume that the node ids $id(v) : v \in V$ are not part of the input; we only know that the ids uniquely identify the nodes but in other respects we assume they were chosen by an adversary in $[1, n]$. Correspondingly, in this model the *symbols* of the routing function s_v are known but s_v itself is not, since the succession of symbols

in the routing function is dependent on the (possibly adversary) assignment of node ids. Correspondingly, the memory at each node must be large enough to store any routing function that may arise over *any* permutation of ids. See Fig. 5.2a.

Even though we do not know the routing function itself, we may still use any statistics on its contents for compression that is invariant to reordering. In the below, we use the next-hop distribution $n_i : i \in [1, \delta_v], v \in V$, where n_i denotes the number of times output port id i appears as a next-hop port in s_v, since this statistics is easy to compute from the routing policy and does not change when reassigning the node ids. Easily, the name-independent model maps to the zero-order model of information-theory, which uses the statistics $n_i/n : i \in [1, \delta_v]$ for compression and thereby does not depend on the assignment of node ids that is beyond our control (Section 5.2.2). In many practically relevant graphs, use of a zero-order compression may yield significant memory savings beyond the information-theoretic lower bound as we show in [124].

The below bounds apply on the attainable compression in the name-independent case.

Theorem 5.2. Given a graph G on n nodes, a node $v \in V$, the next-hop alphabet $\Sigma_v = [1, \delta_v]$, and the routing policy that assigns next-hops at v, let n_i denote the number of times the routing policy assigns output port id $i \in [1, \delta_v]$ as a next-hop to any destination node $u \in V : u \neq v$. Then, storing the routing function s_v requires at least $nH_0(v)$ bits memory, where $H_0(v)$ is the empirical entropy of s_v over any assignment of node ids:

$$H_0(v) = H_0(s_v) = \sum_{i \in [1, \delta_v]} \frac{n_i}{n} \log \frac{n}{n_i} \ . \tag{5.4}$$

In addition, there is an encoding scheme that stores s_v in at most $nH_0(v) + o(n)$ bits and supports lookups in $O(\log n)$.

Proof. Proving that $nH_0(v)$ bits is a lower-bound for storing s_v in the name-independent model is trivial, using basic ideas available in every textbook on data compression [32,48]. We still reproduce the proof briefly below to demonstrate the main idea and to refer the reader back to this proof later when these ideas can be reused in other contexts.

Assuming an adversarial node id assignment, any algorithm storing s_v will need to be able to distinguish all routing functions in which the next-hop port id 1 appears at exactly $\binom{n}{n_1}$ positions, port id 2 appears at $\binom{n-n_1}{n_2}$ positions, etc., putting the number of distinct routing functions to

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

Then, uniquely identifying each possible routing function will need $\log \frac{n!}{\prod_i n_i!} = nH_0(v) + o(n)$ bits, using the Stirling formula: $\ln n! \sim n \ln n - n + O(\ln n)$.

To prove that $nH_0(v) + o(n)$ bits is also an upper bound, we may use essentially any compressed string self-index from [16, 69] (recall Section 5.2.4); e.g., generalized wavelet trees [65] and Huffman-shaped wavelet trees will attain $nH_0(v) + o(n)$ bits of space and $O(\log n)$ random access. □

We note that the above upper bound is for the *static* case, i.e., when there is no need to support fast updates. For the dynamic case, where we want to support updates to the compressed forwarding table without a full re-build, we cannot reach this size at the moment: the best known encodings attain $nH_0(s) + o(n \log \log n)$ bits space and random access and update in $O(\log n \log \log n)$ time [141]. For more details, see [66, 100, 155, 235].

Example 5.2. Consider the running example in Fig. 5.2. The first address space was obtained by assigning ids to nodes by a random shuffle of the integers $[1, n]$, see Fig. 5.2a. The entropy of the corresponding routing function for the thick node (see the figure) is $\bar{H}_0 = 1.1$ bit in the zero-order model, roughly equal to the first-order entropy $\bar{H}_1$ due to the independence of node ids. This setting illustrates the name-independent model where we have no control over the assignment of node ids, and the best we can do is to assume that node ids were set by an adversary, and thereby effectively random.

5.3.4 Name-dependent Case

In the final model we discuss in the context of the flat address space model, 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 the compact routing literature [127] and, as one easily checks, it maps to the higher-order models of information-theory (Section 5.2.3).

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 in the higher-order model can function as a *structured address space* for the network at hand (see the next Section). 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 small higher-order entropy.

The corresponding space bounds are then as follows.

Theorem 5.3. Given a graph G on n nodes, the assignment of node ids $id(u) : u \in V$, a node $v \in V$, the next-hop alphabet $\Sigma_v = [1, \delta_v]$, and the routing policy that assigns next-hops at v, let n_i denote the number of times the routing policy assigns output port id $i \in [1, \delta_v]$ as a next-hop to any destination node $u \in V : u \neq v$ and let n_{qi} be the number of times the routing policy assigns exactly the next-hop sequence $q \in [1, \delta_v]^k$ to the k addresses before an address to which next-hop i is assigned. Then, encoding the routing function s_v needs at least $nH_k(v)$ bits for any $k > 0$ integer, where

$$H_k(v) = H_k(s_v) = \sum_{q \in [1,\delta_v]^k} \sum_{i \in [1,\delta_v]} \frac{n_{qi}}{n} \log \frac{n_i}{n_{qi}} \ . \tag{5.5}$$

In addition, there is an encoding scheme that stores s_v in at most $nH_k(v) + o(n)$ bits for any $k = o(\frac{\log n}{\log \log n})$ and supports lookups in $O(\log n \log \log n)$ time.

Proof. The proof for the lower-bound part is a trivial based in the ideas in the proof of Theorem 5.2, but see also [48]. For attaining the above entropy bounds, one may use the higher-order string indexers in [67, 69]. □

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 the next-hop port, 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. The below example illustrates this strategy.

Example 5.3. Consider again Fig. 5.2. The second address space here was obtained by "de-randomizing" the address space in order to demonstrate the difference of the higher-order model from the zero-order setting, see Fig. 5.2b. This address space was found by a brute force algorithm, which aims to find the permutation π of the node id space $[1, n]$ that minimizes the average first-order entropy of nodes' routing functions:

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

The resultant "optimized" address space is far from being random: in fact, it has the curious property that nodes that are close to each other in the graph are mapped to nearby node ids in the address space, which translates to three-fold space reduction compared to a random address space: $\bar{H}_1 = 0.355 \ll \bar{H}_0 = 1.1$. This setting illustrates the name-dependent model, where we have control over the node ids and hence we can find an address assignment that minimizes the higher-order entropy. See more on address space optimization in [124].

5.4 Forwarding Table Compression in Hierarchical Routing

So far we have discussed forwarding table compression over a "flat" address space. The very purpose to adopt this flat address space was to eliminate intrinsic address space "structure"; this way, the flat node ids allow us to model forwarding tables as simple "unstructured" sequences of symbols (strings) and develop the related space bounds and encoding schemes using basic results from information theory and compressed data structures. Curiously though, as we gradually moved from the simple zero-information model and the name-independent model to the more complex name-dependent model, and the related "optimized" address spaces, we could gradually introduce address space structure in the form of "node id proximity" (recall Example 5.3), and thereby attain higher compression. Nevertheless, the address space structure has remained highly elusive, buried underneath an intricate higher-order information-theoretical argument. In this Section, we make this argumentation explicit and turn to discuss forwarding table compression over the exemplary structured address space: hierarchical routing [121].

5.4.1 Formal Model

Hierarchical routing. The main motivation in hierarchical routing is to curtail the amount of routing state that is stored in network nodes using geographical or topological "graph aggregation" [121]. The idea is to hierarchically partition the network into increasingly smaller groups (or *clusters*) of nodes based on topological proximity and then representing entire clusters with a single entry in the forwarding tables. 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 up to the top-level.

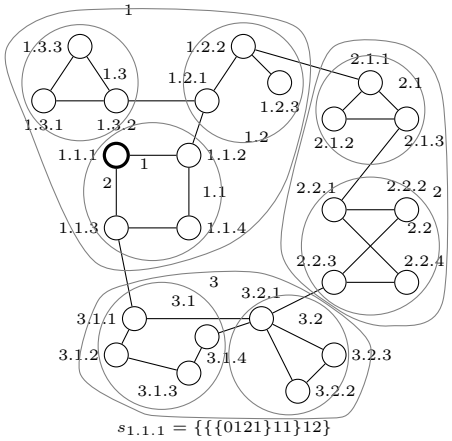


Figure 5.3: Sample graph from Kleinrock's original paper on hierarchical routing [121], with hierarchies and node addresses as of therein.

Example 5.4. Fig. 5.3 reproduces the hierarchical clustering from Kleinrock's original paper [121] on the running example Fig. 5.2. Here, node `1.1.1` maintains a separate entry for each node in its own cluster (i.e., all nodes with addresses `1.1.*`), one entry per each second-level cluster (addresses `1.*`) in its own top-level cluster, and one for each top cluster. Easily, the address space is now hierarchical, where a node's location in the graph is fully described by the succession of increasingly larger higher-level clusters that contain it (plus its own id), rendering the address space structure explicit that was only implicit in flat addressing.

Forwarding table compression is implicit in hierarchical routing, in that just by the act of aggregating entire groups of nodes into a single cluster and representing the entire cluster with a single entry in the forwarding table already delivers substantial routing state space reduction. On top of this, we can usually attain further compression using a sophisticated prefix tree compression algorithm, as we show below. This state aggregation does not come for free, however: as clusters are aggregated higher up in the cluster hierarchy we gradually lose the precise information on the small-scale topology of low-level clusters, which may then lead to suboptimal routing.

Example 5.5. Consider again Fig. 5.3 and assume shortest-path routing. Here, node `1.1.1` maintains a single entry with respect to each node with address in `2.*.*` even though, for strict shortest-path routing, it would need to maintain separate forwarding table entries to reach `2.1.1` and, say, `2.2.3`, because the corresponding shortest paths take different next-hop ports from `1.1.1`.

Consequently, state aggregation in hierarchical routing necessarily incurs path-length increase. This is in sharp contrast to the name-dependent model on flat addresses (Section 5.3.4), where our forwarding table compression schemes were required to precisely reproduce the optimal paths (as per the routing policy). In hierarchical routing suboptimality is not only allowed, it is *inherent* in the routing model. Fortunately, the resultant path-length growth (usually referred to as *path inflation* in the related literature [196]) is not significant: in the case of the Internet, which relies on hierarchical routing (see below), empirical studies confirm that path dilation affects only a limited fraction of source-destination pairs and the resultant stretch remains under 120–150% [50, 196].

IP forwarding tables. In his seminal paper [121], Kleinrock has laid down the conceptual foundations for a scalable routing architecture for large-scale networks. However, it was the ARPANET, and later the Internet and the Internet Protocol (IP) suite, which has turned the conceptual idea to tangible reality that is still operational, 40 years after the original paper. Below, we develop a formal model for representing IP forwarding tables and then we define the corresponding compression schemes. In what follows we shall concentrate on version 4 of the IP suite exclusively (IPv4); the results transform to IPv6 with trivial modifications.

Clusters, as per hierarchical routing, are called *subnets* in IP routing. Node ids in IPv4 are 32-bit unsigned integers and a subnet in this address space is defined by a *subnet prefix* `X/Y` (in the so called CIDR notation), where Y is prefix-length and X is the $32-Y$ bit subnet identifier (counting from the MSB in network-byte order) usually denoted in the "dotted decimal notation". This way, the prefix `0.0.0.0/0` (or simply `0/0`) covers the entire IPv4 address space and, oppositely, a subnet with prefix length 32 identifies a single IP address. In addition, a prefix `A/B` contains the *more specific* prefix `C/D` if `A` is a prefix of `C` of length $B < D$. A prefix *matches* an IP address if the IP address is a (fully-specified) more specific subnet for the prefix. For more detail on IP addressing, the reader is referred to the textbook [26].

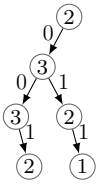
The IP routing model implements a slightly modified version of hierarchical routing. Similarly to hierarchical routing the IP forwarding table represents the entire subnet with a single entry, but whereas in hierarchical routing the forwarding table stores each remote cluster at most once, in IP routing we allow a subnet to be specified multiple times; in particular, not just the subnet prefix but all the "less specific" subnets that "contain" it as a more specific may also be explicitly stored, with a distinct next-hop label (see Example 5.6 below). Disambiguation in such cases occurs by the *longest prefix matching* (LPM) rule: whenever multiple subnet prefixes match an IP address then the forwarding table lookup algorithm must return the most specific matching prefix, with the corresponding next-hop label. Efficiently supporting this complex longest-prefix-matching semantics is perhaps one of the most difficult aspects of IP forwarding.

Example 5.6. A sample IP forwarding table is given in Fig. 5.4a. This table stores for a set of subnet prefixes (i.e., for different clusters as per hierarchical routing) the corresponding output port along the preferred path in the form of an opaque next-hop label. Whereas in hierarchical routing the forwarding table stores each cluster at most once, in IP routing we allow a subnet to appear multiple times in the forwarding table; e.g., for the subnet `96.0.0.0/3` we have a direct entry as well as a separate entry for its immediate "parent" less specific subnet `64.0.0.0/2`, the grand-parent `0.0.0.0/1`, and the top-level subnet `0.0.0.0/0`. Then, the IP address `96.0.0.1` (binary form `011...`) is matched by all these prefixes and the longest matching subnet prefix is `96.0.0.0/3`, assigning the next-hop label 1 to any packet received with this IP address as destination; for `79.120.55.19` (binary form `010...`) the LPM result is `64.0.0.0/2` and the lookup result is the corresponding next-hop label 2; and finally for `152.66.240.111` (binary `1...`) the only matching prefix is the top-level prefix `0.0.0.0/0` and so packets destined to this address are to be forwarded to the "default gateway", identified by the next-hop label 2.

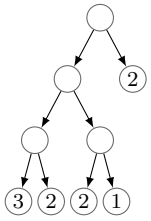
Let N denote the number of entries in the FIB and let $\delta = |\Sigma|$ denote the number of next-hops. An IP router does not keep an adjacency with every other router in the Internet and so $\delta \ll N$; specifically we assume that δ is $O(\text{polylog}\, N)$ or $O(1)$ [44, 207].

CIDR prefix	binary	label
0.0.0.0/0	-/0	2
0.0.0.0/1	0/1	3
0.0.0.0/2	00/2	3
64.0.0.0/2	01/2	2
32.0.0.0/3	001/3	2
96.0.0.0/3	011/3	1

(a)



(b)



(c)

i	S_I	S_α	
1	0	2	level 0
2	0	3	level 1
3	1	2	
4	0	2	level 2
5	0	1	
6	1	-	
7	1	-	level 3
8	1	-	
9	1	-	

(d)

Figure 5.4: Representations of an IP forwarding table: (a) tabular form with subnet address in CIDR notation, binary format, and next-hop address label; (b) prefix tree with edge labels marked; (c) leaf-pushed trie; and (d) *XBW-b* transform (see later).

Finally, let W denote the width of the address space in bits (e.g., $W = 32$ for IPv4 and $W = 128$ for IPv6).

Trivially, the tabular representation of IP forwarding tables (see Fig. 5.4a) is not the most efficient one. The storage size is $(W + \lg \delta)N$ bits (recall Observation 4.7 from the previous Chapter), but what is worse a single lookup or update operation would require looping through the entire table, taking $O(N)$ time, which is clearly prohibitive when IP routers regularly store almost a million prefixes ($N \sim 10^6$) and support hundreds of millions of lookups per second. This makes IP packet forwarding one of the most compelling use cases for compressed data structures.

Prefix trees. For hierarchical routing, and for IP forwarding in particular, binary prefix trees (or *tries* [190]) support lookup and update much more efficiently than the tabular form. A trie is a labeled ordinal tree, in which every path from the root node to a leaf corresponds to an IP prefix and lookup is based on successive bits of the destination address: if the next bit is 0 proceed to the left sub-trie, otherwise proceed to the right, and if the corresponding child is missing then return the last label encountered along the way. Prefix trees generally improve the time to perform a lookup or update from linear to $O(W)$ steps, at the cost of increasing memory size to roughly $O(N \log N)$ bits.

Example 5.7. The prefix tree for the sample IP forwarding table is given in Fig. 5.4b. The LPM search for the IP address `79.120.55.19` (binary form: `010...`) goes as follows: we start from the root and we store the node's next-hop label 2 in a temporary variable, take the left child along the edge marked with the label 0 since the first bit is `0` to arrive to a node labeled with 3, we overwrite the temporary variable storing the best match found so far to 3, then we take the right child storing label 2 again, and then we notice that the left child is missing so we terminate the search and return the last label found, i.e., 2. Similarly, the LPM result for `152.66.240.111` (binary `1...`) is next-hop label 2 as found in the root node since the LPM search terminates at the first (missing) child.

Obtaining a prefix-free form. There is a considerable literature on compressed representations for *general* labeled trees and related notions of entropy, see a recent survey in [67]. Unfortunately, none of these proposals can be used readily for IP forwarding table compression due to the special semantics of IP lookups. In particular, whereas in standard labeled tries and search trees it is enough to return the (single) label identified by the search key, in IP forwarding the longest-prefix-matching semantics requires to either return *all* matching prefixes or, better yet, make an additional search on less specifics to

find the smallest prefix that contains the search key (i.e., the IP address). The major difficulty in IP forwarding table compression is, therefore, to modify a well-known trie compression scheme from the literature in a way as to support LPM lookups and, at the same time, controlling the analytical storage size bounds.

The main idea in our schemes is to explicitly "disambiguate" the prefix tree prior to subjecting it to information-theoretical analysis and compression, in a way that *forwarding equivalence* is maintained: the disambiguated prefix tree will return exactly the same result for each LPM search as the original IP forwarding table would do but it will possess a certain set of properties that will make it much simpler to compress.

In order to convert the prefix tree into a unique, normalized form we adopt the *leaf-pushing* technique. In leaf-pushing, a preorder traversal first pushes labels from the parent nodes towards the leaves and then, in a second postorder traversal, each parent with identically labeled leaves is substituted with a leaf marked with the children's label.

Next, we introduce some terminology that will be useful later.

- A *proper* binary trie is such that it possesses the following nice structure: any node is either a leaf node or it is an interior node with exactly two children. A leaf-pushed trie is a proper trie.
- A *leaf-labeled* trie has the property that interior nodes do not maintain labels while all leaves are explicitly labeled. Leaf-pushed tries are leaf-labeled.
- By the previous property there are no "less specifics" in the leaf-pushed trie, implying that the transformed IP forwarding table is *prefix-free*.
- The leaf-pushed trie is *forwarding-equivalent* with the original trie; for proofs see our earlier work [120], also [60, 193, 197].
- Given a leaf-labeled trie, the string composed by enumerating the leaf labels in breadth-first-search order from left to right is called the *label map* of the trie.

Example 5.8. The leaf-pushed prefix tree for the sample IP forwarding table is given in Fig. 5.4c. Observe that the leaf-pushed trie is proper, leaf-labeled, and prefix-free; in addition, the label map of the trie is given by the string "`23221`" (see S_α in Fig. 5.4d). The LPM search goes similarly as before, always tracking the edge whose label coincides with the subsequent bit of the IP address, but this time we do not need to memorize temporary lookup results since, due to the edge-labeling property and by the prefix tree being proper we are guaranteed to terminate in a labeled leaf node whose label we immediately return. For instance, for the IP address `79.120.55.19` (binary form: `010...`) we land at the downmost third leaf counted from the left, marked with the next-hop label 2. Observe that this LPM result is exactly the same as in the tabular form and in the original prefix tree; this illustrates that leaf-pushing yields a forwarding equivalent representation.

5.4.2 Compressing IP Forwarding Tables

The main reason why we introduced the leaf-pushed representation is that, by its "nice" properties, it yields a labeled prefix tree form in which LPM search boils down to a simple tree traversal, just like on any ordinary search tree. This observation then opens up the opportunity to leverage existing information-theoretical analysis, and the corresponding compressed encoding schemes from the literature, to reduce the size of IP forwarding tables.

In the rest of this Section, we prove our main result on compressing forwarding tables in hierarchical routing in general, and in IP forwarding in particular.

Theorem 5.4. Suppose we are given an IP forwarding table T as a proper, binary, leaf-labeled trie T with n leaves and label map s defined on the label alphabet $\Sigma = [1, \delta]$. Then, storing T requires at least $2n + nH_0(s)$ bits of memory, where $H_0(s)$ is the zero-order empirical entropy, as of (5.4), of the label map s. In addition, there is an encoding scheme that stores T in at most $2n+nH_0(s)+o(n)$ bits so that `lookup` on the compressed representation terminates in $O(W)$ time if $\delta = O(\text{polylog}\, n)$.

In the rest of this Section, we prove the Theorem through a series of technical lemmas.
Entropy bounds. First, we show that $2n + nH_0(s)$ bits is a lower bound for storing T.

Lemma 5.5. Let T be a proper, binary, leaf-labeled trie with n leaves and label map s. Then, storing T requires at least $2n+nH_0(s)$ bits of memory, where $H_0(s)$ is the empirical entropy of the label map s as given by (5.4).

Proof. The lower bound is justified with a counting argument. The number of proper binary trees on n leaves is given by the $(n-1)$-th Catalan number $C_{n-1} = \frac{1}{n}\binom{2n-2}{n-1}$, therefore we need at least $\lg C_{n-1} = 2n - \Theta(\log n)$ bits to encode the tree itself [67, 106]. Storing the label map defined on the n leaves of T requires an additional $nH_0(s)$ bits, using the same arguments as in the proof of Theorem 5.2, and since the tree and the label map are independent we get the required result. □

Intuitively speaking, the entropy of the tree structure corresponds to the information-theoretic limit of $2n$ bits as we do not assume any regularity in this regard. To this, the leaf-labels add an extra nH_0 bits of entropy.
Encoding scheme. Next, we present our compressed IP forwarding table data structure that we call the *Burrows-Wheeler transform for binary leaf-labeled tries* (*XBW-b*). This data structure combines several existing schemes into a single versatile IP forwarding table representation; namely, the tree compressor comes from Jacobson [106] and the Burrows-Wheeler transform from [67], but we adopted these structures from general labeled trees to leaf-labeled binary tries; and see also [174, 180].

Let T be a proper, leaf-labeled binary prefix tree, let t be the number of nodes in T, let L be the set of leaves, let $n = |L|$, and let l be a mapping $L \mapsto \Sigma = [1, \delta]$ specifying for a leaf v the corresponding next-hop label $l(v) \in \Sigma$. The following claims hold:

P1: Either $v \in L$, or v has exactly two children.

P2: $t = 2n - 1 = O(n)$.

The main idea in *XBW-b* is serializing T into a bitstring S_I that encodes the tree structure and a string S_α on the alphabet $\Sigma = [1, \delta]$ that encodes the labels, and then using a lossless string compressor to obtain the storage size bounds. Correspondingly, the *XBW-b* transform is defined as the tuple $\text{xbwb}(T) = (S_I, S_\alpha)$, where

- S_I is a bitstring of size $t = 2n - 1$ with zero in position i if the i-th node of T in level-order is an interior node and 1 if it is a leaf; and
- S_α is the leaf-map, i.e., the string of size n on the alphabet Σ that encodes the leaf label $l(v)$ for each $v \in L$.

Example 5.9. For our sample FIB, the leaf-pushed trie and the corresponding *XBW-b* transform are given in Fig. 5.4c and Fig. 5.4d, respectively.

Construction and IP lookup. In order to generate the *XBW-b* transform, one needs to fill up the strings S_I and S_α, starting from the root and traversing T in a breadth-first-search order. Meanwhile, two counters are maintained: i is used to enumerate the nodes and index into S_I, while j counts the leaves and indexes S_α. For every node we decide whether it is an interior node, in which case the corresponding entry of S_I is set to zero, or it is a leaf and so S_I is set to 1 and the leaf label is appended to S_α.

```
1: i ← 1; j ← 1
2: BFS-TRAVERSE (node v, integer i, integer j)
3:     if v ∉ L then S_I[i] ← 0
4:         else S_I[i] ← 1; S_α[j] ← l(v); j ← j + 1
5:     i ← i + 1
6: end BFS-TRAVERSE
```

The following statement is now obvious.

Lemma 5.6. Given a proper binary, leaf-labeled trie T on n leaves, xbwb(T) can be built in $O(n)$ time.

The transform xbwb(T) has some appealing properties. By being based on a breadth-first-search traversal, the children of some node, if exist, are stored on consecutive indices in S_I and S_α. In fact, *all* nodes at the same level of T are mapped to consecutive indices.

The next step is to actually compress the strings. This promises easier than compressing T directly as xbwb(T), constituted by two sequential string representations, lacks the intricate structure of tries. An obvious choice would be to apply some standard string compressor (like the venerable `gzip(1)` tool), but this would not admit navigation queries like "get all children of a node" without first decompressing the transform. Thus, we rather use a compressed string self-index [67, 68, 106, 166] to store xbwb(T), which facilitates efficient navigation immediately on the compressed form.

The way string indexers usually realize navigability is to implement a certain set of simple primitive in-place operations. Given a string $S[1, t]$ on the alphabet Σ, a symbol $s \in \Sigma$, and integer $q \in [1, t]$, these primitives are as follows:

- access(S, q): return the symbol at position q in S;
- rank$_s$(S, q): return the number of times symbol s occurs in the prefix $S[1, q]$; and
- select$_s$(S, q): return the position of the q-th occurrence of symbol s in S.

Curiously, these simple primitives, if implemented in optimal time (e.g., in $O(1)$ or $O(\log n)$ [16, 65, 69]), admit strikingly complex queries to be implemented and supported efficiently. In particular, the IP lookup routine on xbwb(T) takes the following form.

```
1: lookup (address a)
2:     q ← 0, i ← 1
3:     while q < W
4:         if access(S_I, i) = 1 then
5:             return access(S_α, rank_1(S_I, i))
6:         r ← rank_0(S_I, i)
7:         f ← 2r
8:         j ← bits(a, q, 1)
9:         i ← f + j
10:        q ← q + 1
11:    end while
```

12: **end** `lookup`

The code walks through the successive bits of the input address (line 3). In each iteration, first it checks if the actual node, encoded at index i in xbwb(T), is a leaf node (line 4). If it is, then $\text{rank}_1(S_I, i)$ specifies how many leaves were found in the course of the BFS-traversal *before* this one and then the corresponding label is returned from S_α (line 5). If, on the other hand, the actual node is an interior node, then r gives the number of interior nodes *before* this one (line 6). As one easily checks, in a level-ordered tree traversal the children of the r-th interior node are encoded from position $2r$ consecutively [106], thus f gets the index of the first child of the actual node (line 7). Next, we obtain the index j of the child to be visited next from the input address (line 8), we set the current index to $f + j$ (line 9), and then we carry on with the iteration.

Memory size bounds. Next, we show that *XBW-b* encodes a leaf-labeled trie to entropy-constrained size.

Lemma 5.7. Let T be a proper, leaf-labeled binary prefix tree with n leaves on an alphabet of size $O(\text{polylog}\, n)$, and let H_0 denote the empirical entropy of the label map. Then, xbwb(T) can be stored on at most $2n + nH_0(S_\alpha) + o(n)$ bits so that `lookup` on xbwb(T) terminates in $O(W)$ time.

Proof. Encode S_I on $2n + o(n)$ bits using the `RRR` succinct bitstring index [166] (or our enhanced version called `R3D3` in slightly smaller space [151]), which supports select and rank in $O(1)$. In addition, S_α can be stored on $nH_0 + o(n)$ bits using generalized wavelet trees so that access is $O(1)$ under the assumption that the alphabet size is $O(\text{polylog}\, n)$ [68] or in $O(\log n)$ otherwise, or the Huffman-shaped wavelet tree construction [16] with access in $O(\log n)$. The xbwb `lookup` routine calls the rank and access primitives on S_I at most once in each iteration, yielding $O(W)$ complexity for the maximum possible W iterations, and the access primitive on S_α at most once, for another $O(1)$ steps for [68] (or $O(\log n) = O(W)$ for [16] or $\delta = \Omega(\text{polylog}\, n)$), to obtain $O(W)$ complexity in total. □

Finally, we note that above zero-order entropy bounds can be easily strengthened to *higher-order* entropy; see [174]. The particular bounds are $2n + nH_k(S_\alpha) + o(n)$ bits space, where H_k is the k-th order entropy of the label distribution for any $k = o(\log_\delta n)$, and lookup in $O(W)$ time.

5.5 The Relation of Entropy Notions

The most prominent large-scale network of our days is the Internet. As we have seen, similarly to Kleinrock's hierarchical routing scheme the address space of the Internet 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-matching semantics.

So far, we have seen two, seemingly unrelated approaches to deal with this structure in IP forwarding table compression. First, there is the "natural" prefix tree representation as above, where address space structure is encoded in the "tree" index S_I. The other approach, based on Section 5.3.4, models the IP address space as a flat id space (of size 2^{32} in the case of IPv4), defines a forwarding table on top of this flat id space as a simple string, and represents "structure" in the address space using the following higher-order argumentation: (1) on the Internet, geographic proximity strongly correlates with proximity in the address space, in that endhosts close to each other will often be numbered

from the same subnet; (2) if two hosts belong to the same subnet then there is a very good chance that they will be assigned the same next-hop label in the IP forwarding table at other nodes (but not 100% due to that more specific prefixes may selectively cover parts of the subnet); (3) therefore the symbol in the forwarding table corresponding to an endhost can be reliably predicted from its context, i.e., the next-hop symbol for endhosts in the same subnet; and finally (4) by the foregoing arguments the more "structure" there is in the address space the smaller the higher-order entropy of the forwarding tables. This idea is perhaps best captured by the following (in)famous principle of hierarchical routing that is often referred to as Rekhter's Law [148]:

"Addressing can follow topology or topology can follow addressing. Choose one."

Below, we argue that these two notions, and the related entropy bounds, are not unrelated; in other words, representing address space structure in the hierarchical routing model can be either explicitly encoded into a prefix tree, or it can be represented as the higher-order statistics of the forwarding table "string representation"; strikingly, as our next result shows the two representations are storage-space-wise identical up to small error terms.

Let T be the proper, leaf-labeled binary prefix tree representation of an IP forwarding table, let W be the depth, let n be the number of leaves in T, let δ denote the number of distinct next-hop labels in T, and let M_T be the minimum space bound 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, and let M_S be the compressed size of S.

We show that the following holds for M_T and M_S.

Theorem 5.8. $M_T \leq M_S \leq KM_T$ for some $1 \leq K \leq 1 + \frac{1}{2}\log\frac{2^W}{n}$.

Proof. We know that 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 some proper K constant.

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

$$\begin{aligned} 2^W H_1(S) &\leq \sum_i (l_i - n_i) \log \frac{l_i}{l_i - n_i} + \sum_i n_i \sum_j \frac{n_{ij}}{n_i} \log \frac{n_i}{n_{ij}} \\ &\leq n \log e + n \sum_i \frac{n_i}{n} \log \frac{l_i}{n_i} + nH_1(T) \leq n \left(\log e + \log \frac{2^W}{n} + H_0(T) \right) \ , \end{aligned}$$

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 $K = \frac{M_S}{M_T} \leq \frac{H_0 + \log e + W - \log n}{H_0 + 2}$. □

Table 5.2: Compressed size of some real IPv4 forwarding tables samples from [180], using the compressed prefix tree representation as per Theorem 5.4 (M_T) and the higher-order compressed string representation as per Theorem 5.3 (M_S).

	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

For a typical real IPv4 forwarding table, $H_0 \sim O(1)$, $n \sim 150,000$, $W = 24$ (very few prefixes are more specific than 24, see [180]), we get $K \sim 3$. Similar argumentation for IPv6 yields $K \sim 7$. Consequently, the size of T is intimately related to the size of S and so our compressed string-representation very closely models trie-based IP forwarding tables. Table 5.2 confirms this finding empirically, by showing the memory size bounds for a set of real-life IPv4 forwarding tables from [180] with $W = 24$. The results indicate that the real value of K might be closer to 1.1–1.5. Strikingly, both our representations compress to very small size (below 300 Kbytes), which is 1-3 orders of magnitude improvement from earlier work [15, 56, 59, 61, 90, 97, 104, 159, 197, 215, 223, 231].

5.6 Related Work

In line with the unprecedented growth of the routed Internet and the emerging scalability concerns thereof [101, 119, 233], finding efficient IP forwarding table representations has been a heavily researched question in the past. Judging from the sheer quantity of recent work [95, 138, 211, 231], the problem of scaling IP forwarding tables does not seem to have been solved completely yet.

The major challenges are (1) the steadily growing routed IP address range (recall Fig. 5.1), (2) the intrinsic complexity of IP lookups, requiring specialized algorithms to support the longest prefix matching semantics, and (3) the vast increase in line rates modern routers have to support efficiently. These days, an operational IP router needs to support hundreds of millions of LPM lookups per second over an IP forwarding table that contains more than 750,000 prefixes, and counting. Note that IP lookup cost is per packet and should be optimized to the extreme to meet wire speed. A way to achieve that is to store the forwarding table in on-chip fast memory, such as CPU caches or FPGA block RAM, which is limited in size and very expensive but supports about 10 times faster random access than off-chip DRAM. This creates a very strong motivation to compress IP forwarding tables to the smallest size possible, while still supporting efficient LPM lookups on the compressed representation.

The idea to store IP forwarding tables in prefix trees dates back to the BSD kernel implementation of Patricia trees [190]. This representation consumes a massive 24 bytes per node, and a single IP lookup might cost 32 memory accesses. Storage space and search time can be saved on by expanding nodes' strides to obtain a multibit trie [40], see e.g., controlled prefix expansion [104, 197], level- and path-compressed tries [159],

Lulea [56], Tree Bitmaps [61] and successors [15, 194, 227], etc. Early attempts to shrink the routing table focused on cleverly relabeling the prefix tree to contain the minimum number of entries (see ORTC and derivatives [60, 211]). An opposite approach presented in [227] is to deliberately *increase* (or "explode") the IP forwarding table to support fast lookups, by splitting it into a constant size on-chip data-structure that supports finding the longest prefix very efficiently and a much larger, but less performance-sensitive, off-chip data structure that then provides the next-hop for the found prefix. Further forwarding table representations include hash-based schemes [15, 215], dynamic pipelining [97], CAMs [143], Bloom-filters [59], binary search trees and search algorithms [90, 231], massively parallelized lookup engines [95, 231], forwarding table caching [138], and different combinations of these (see the text book [223]).

Curiously, none of these prior works comes with information-theoretic space bounds, or any sorts of analytic storage size characterization apart from, usually overly loose and largely prohibitive, worst-case bounds. Although next-hop entropy itself appears in [211], but no analytical evaluation ensued. This is worrisome, since unknown algorithmic corner-cases and runaway space-time complexity may result that the network infrastructure built on top of these schemes may exhibit certain hard-to-explain performance regressions in unexpected situations. Apart from an operational network debugging nightmare, this unlocks yet unseen cyberphysical threats, as a malicious user may easily exploit these algorithmic deficiencies to launch denial of service attack on a communication network. We show such an attack in our recent work [51], where we exploit an algorithmic corner case in the most popular SDN switch, Open vSwitch, whereby we send a carefully fine-tuned mix of random packets to drive the switch into a state where the space-time complexity of the internal forwarding table data structure skyrockets, slowing the switch from the normal gigabits per second speed to a mere 10–100 kilobits per second.

In contrast, our IP forwarding table compression schemes come with *theoretically proven* space limits, and thus *predictable memory footprint and lookup complexity* under any workload. Crucially, our space characterizations are not of generic worst-case nature, like in the previous Chapter where unscalability meant that there is *at least one* graph where a routing policy is incompressible, but rather our framework gives specific bounds for particular inputs, graphs, routing policies, address spaces, etc. Our compression algorithms are, therefore, *opportunistic* [64], in the sense that whenever a problem instance admits compression our algorithms will attain maximum possible space reduction subject to stringent information-theoretic analysis, which may go way beyond what a *generic* IP forwarding table encoding scheme would allow.

The latest reported forwarding table size bounds for >400K prefixes range from 780 KBytes (DXR, [231]) to 1.2 Mbytes (SMALTA, [211]) and > 24 Mbytes [76, 159]. Our IP forwarding table compression scheme, *XBW-b*, improves this to just 100–300 Kbytes, which easily fits into today's SRAMs or can be realized right in chip logic with modern FPGAs.

These improvements have been made possible by the recent advances on compressed data structures in theoretical computer science research [55, 64, 68, 100, 141, 154, 155, 166, 235]. A compressed data structure not only provides compression to entropy limited size for the underlying data, but also provides quick operations (like access, rank, select) on the compressed form in place. Jacobson in his seminal work [106] defined succinct encodings of trees that support navigational queries in optimal time within information-theoretically limited space. He was also the first to recognize the importance of the select

and rank primitives and pointerless data structures. Jacobson's bitmap-based techniques later found important use in forwarding table aggregation [15,61,194]. With the extensive use of bitmaps, *XBW-b* can be seen as a radical rethinking of these schemes, inspired by the state-of-the-art in succinct and compressed data structures.

The results presented in this Chapter constitute only a small fraction of the work we have done in the general field of routing scalability research and IP forwarding table compression. Below, we highlight some of our recent results in the field.

One direction was to combine several independently proposed techniques to attain the greatest possible compression on IP forwarding tables. Using higher-order compressors, level-compression, and several novel ideas, we were able to reduce real-life IP forwarding table instances to a mere 70–120 Kbytes, while still supporting fast LPM lookups [174]. Another goal we tackled was to maximize the applicability of forwarding table compression. Two important limitations of *XBW-b* are that it is static, meaning that updates mean full reconstruction, and relatively slow: while pointerless compressed data structures for bitstring encoding are *theoretically* optimal in terms of complexity, the implementations are still not up to the extreme speed requirements in modern IP routers.

Correspondingly, in [120,180,181] we substituted the pointerless data structures used in *XBW-b* for a trie-based approach that yielded unprecedented lookup speed and full dynamism through quick updates. The idea was to essentially re-invent the classic prefix tree, borrowing the basic mechanisms from the Lempel-Ziv (`LZ78`) string compression scheme [32,48]. `LZ78` attains entropy by parsing the text into unique sub-strings, yielding a form that contains no repetitions. Tree threading is a generalization of this technique to *unlabeled trees*, converting the tree into a Directed Acyclic Graph (DAG) by merging isomorphic sub-trees [104,114,193,199]. In [180,181] we took a step further and applied this idea to *labeled trees*, merging sub-tries taking into account both the underlying tree structure *and* the labels [31, 46]. If the trie is highly regular then this will eliminate all recurrent sub-structures, producing a representation that contains no repetitions and hence admits entropy bounds like `LZ78`. The resultant technique is called *trie-folding*. In [180,181] we were able to show that the compressed prefix-DAGs produced by trie-folding satisfy the same IP forwarding table entropy size bounds as *XBW-b* as of Theorem 5.4 up to a small constant factor. In [120], we reduced the memory footprint even further by combining trie-folding with controlled level-compression.

The basic idea of folding a labeled tree into a DAG is not particularly new; in fact, this is the basis of many tree compacting schemes [114], space-efficient ordered binary decision diagrams and deterministic acyclic finite state automata [31], common subexpression elimination in optimizing compilers [46], and it has also been used in forwarding table aggregation [104,193,199] earlier. Perhaps the closest to trie-folding is Shape graphs [193], where common sub-trees, without regard to the labels, are merged into a DAG. However, this necessitates storing a giant hash for the next-hops, making updates expensive especially considering that the underlying trie is leaf-pushed. Trie-folding, in contrast, takes labels into account when merging and also allows cheap updates. What is more, our prefix-DAGs provably attain a entropy-constrained size *besides* fast operations, and in this regard our research is still unparalleled until today.

Chapter 6

Summary of New Results

The final Chapter in this Dissertation is devoted to re-state the results scattered throughout the earlier chapters, but this time in a structured form.

The main theme of the Dissertation is interdisciplinary approaches to classical open problems in network communications. Accordingly, in the below summary the results will be organized by the problem domain addressed. For each problem, the main contribution is claimed first and then follow the individual "sub-claims".

6.1 Fairness in Internet Routing: The Geometric Perspective

The first main result closes a decade-old problem first raised in [133, Section "When bottleneck and water-filling become less obvious"]. Our approach is interdisciplinary: applying *geometric* arguments to a problem that was formerly addressed in a graph-theoretic framework allows us to get surprisingly deep new insights into the structure of the problem. Consult Table 2.1 for the notation used below.

Thesis 1. *I defined the general max-min fair bandwidth allocation problem as an extension of the classical fixed-path max-min fair bandwidth allocation problem where the forwarding path of each user is not part of the input. Using a geometric approach, I gave a tight characterization for the set of feasible allocations in the generalized problem, I provided a interpretation of different notions of fairness in the geometric model, I extended the bottleneck argumentation from the fixed-path model to the general setting and identified certain "bottleleck cuts" as the equivalent of "bottleneck links" in the fixed-path modelm and I gave a generalized water-filling algorithm to compute it that runs in polynomial time provided the input is also of polynomial size.*

The first claim characterizes the set of feasible bandwidth allocations in a capacitated network G_c. For the full version see Theorem 2.8, also [169–171, 173].

Thesis 1.1. *For any network configuration G_c, the feasible set $T(G_c)$ of the generalized rate allocation problem is a convex polyhedron. Additionally, if G_c is regular then $T(G_c)$ is bounded, full-dimensional, and down-monotone. In general the size of the half-space representation of $T(G_c)$ may grow exponentially with the network size (irrepresentability).*

As a weak form of fairness, the second claim characterizes the so called "non-dominated" bandwidth allocations, which have the property that the allocation of a specific user can

be increased only at the price of decreasing the allocation of another user. For a precise definition of what we mean by "fairness" in this context see Section 2.2; and for the full version of the claim see Theorem 2.12, also [169–171, 173].

Thesis 1.2. *Consider a feasible rate allocation $\theta \in T(G_c)$ and let $\mathcal{N} \subseteq \mathcal{K}$ denote the set of non-dominated users at θ. Then, $\mathcal{N} \neq \emptyset$ if and only if there exists an inequality $\beta^T\theta \leq b$ that is (i) valid, i.e., for each $\theta' \in T(G_c) : \beta^T\theta' \leq b$; (ii) tight, i.e., $\beta^T\theta = b$; and (iii) complementary: $(\beta)_k > 0$ if and only if $k \in \mathcal{N}$.*

As a much stronger fairness notion, the following claim gives a characterization for Pareto-efficient allocations for which *all* users are non-dominated. For the full version of the claim see Observation 2.13 and Theorem 2.14, also [169–171, 173].

Thesis 1.3. *A rate allocation $\theta \in T(G_c)$ is Pareto-efficient, if and only if there exists an inequality $\beta^T\theta \leq b$ that is (i) valid, i.e., for each $\theta' \in T(G_c) : \beta^T\theta' \leq b$; (ii) tight, i.e., $\beta^T\theta = b$; and (iii) strictly positive: $\forall k \in \mathcal{K} : (\beta)_k > 0$.*

Finally, a max-min fair allocation is such that, loosely speaking, "there is no way to make any person better off without hurting anybody else who is already poorer"; for a precise definition, see Definition 2.5. The existence and uniqueness of the max-min fair allocation is stated in Corollary 2.11. The next result gives the characterization as in the previous cases; for the full version see Theorem 2.15, also [169–171, 173].

Thesis 1.4. *A rate allocation $\theta \in T(G_c)$ is max-min fair, if and only if for each user $k \in \mathcal{K}$ there exists an inequality $\beta^T\theta' \leq b$ that is (i) valid, i.e., for each $\theta' \in T(G_c) : \beta^T\theta' \leq b$; (ii) tight, i.e., $\beta^T\theta = b$; and (iii) max-min complementary: $\forall l \in \mathcal{K} : (\beta)_l > 0$ if and only if $\theta_l \leq \theta_k$.*

The water-filling algorithm is a commonplace iterative polynomial-time algorithm to obtain a max-min fair bandwidth allocation in the fixed-path model; see the classic textbook [26]. The below claim merely states the correctness of a natural generalization of this algorithm to the routing-independent version; the algorithm itself is given in Algorithm 2.1 and the claim is given in full as Corollary 2.16, see also [169–171, 173].

Thesis 1.5. *The generalized water-filling algorithm given by Algorithm 2.1 is correct to obtain a max-min fair allocation over $T(G_c)$.*

The algorithm runs in polynomial time provided that the input, in particular the throughput polytope, is of polynomial size. This, however, is not guaranteed in general due to the irrepresentability of the throughput polytope; see Theorem 2.8 and also [29].

The final result translates the bottleneck argumentation from the geometric setting, where bottlenecks were represented as *valid inqeualities* or half-spaces, into the realm of graph-theory. Namely, this concluding result will represent bottlenecks as certain *bottleneck cuts* of the graph: the cut is a "bottleneck" as the links of the cut are guaranteed to be filled to capacity no matter how we realize the max-min fair allocation (by a routing) and also "max-min fair" in a very specific sense (see below). The full exposition of the idea is given in Section 2.4.2, the claim itself is given in Theorem 2.19, see also [169–171, 173].

Thesis 1.6. *Given a network configuration G_c, an allocation of rates $\theta \in T(G_c)$ is max-min fair if and only if for each user $k \in \mathcal{K}$ there is a cut $\mathcal{C}_k$ in G_c so that*

- $\forall\ (i,j) \in \mathcal{C}_k : \sum_{l \in \mathcal{K}_{\mathcal{C}_k}} \sum_{P_q \in \mathcal{P}_l : (i,j) \in P_q} u_q = c_{ij}$ *for any routing* $u \in M(G_c)$ *that realizes* θ *and*
- $l \in \mathcal{K}_{\mathcal{C}_k} \Leftrightarrow \theta_l \leq \theta_k$,

where $\mathcal{K}_{\mathcal{C}_k} \subseteq \mathcal{K}$ *denotes the set of users whose source is separated away from the respective destination by* $\mathcal{C}_k$.

6.2 Adaptive Routing: The Control-theoretic Perspective

The second main result addresses the problem of multipath rate-adaptive routing and approaches the problem in a novel interdisciplinary setting, by casting it in a *control-theoretic* framework instead of the conventional flow-theoretical arguments. This model allows to design general routing controllers that do not rely on static or estimated traffic matrices but rather dynamically adapt to varying demands. Consult Table 2.1 for the notation used below.

Thesis 2. *I formulated the rate-adaptive multipath routing problem in the framework of constrained optimal control. I showed that for any network there exists an optimal rate-adaptive multipath routing algorithm that is (i) stable, (ii) optimal with respect to any optional linear or quadratic objective function, and (iii) feasible, in that it can accommodate any admissible traffic matrix in the network without violating link capacities. Finally, I gave a new complexity characterization for optimal routing controllers.*

The control-theoretic model is specified in Section 3.2.3: given a network configuration G_c, the ORAR model is defined by the plant:

$$x_k(t+1) = x_k(t) - \tau \sum_{P \in \mathcal{P}_k} u_P(t) \qquad \forall k \in \mathcal{K} \qquad \text{(D)}$$

$$x_k(0) = \theta_k \qquad \forall k \in \mathcal{K} \qquad \text{(I)}$$

the constraints (C):

$$\sum_{k \in \mathcal{K}} P_k u_k(t) \leq c \qquad \text{(C1)}$$

$$u_k(t) \geq 0 \qquad \forall k \in \mathcal{K} \qquad \text{(C2)}$$

$$x_k(t) \geq 0 \qquad \forall k \in \mathcal{K} \qquad \text{(C3)}$$

and the payoff (P):

$$\min P(u(.), x(0)) = q_f^T x(N) + \sum_{t=0}^{N-1} r^T u(t) + q^T x(t)\ . \qquad \text{(P)}$$

The first result states that this system is "well-behaved" in a control-theoretic sense; see Lemma 3.4 for the full claim, also [157, 158, 177, 178].

Thesis 2.1. *If a network configuration* G_c *is regular and* $q_f^T > 0$, $r^T \geq 0$, *and* $q^T \geq 0$, *then the system is both controllable and observable under the ORAR model.*

The second result guarantees the existence of a control law under the ORAR model. For a full version of the result see Lemma 3.5, also [157, 158, 177, 178].

Thesis 2.2. *Given a regular network configuration G_c, consider the ORAR model defined by the plant (D)–(I), the constraints (C), and the payoff (P), and suppose that $q_f^T > 0$, $r^T \geq 0$ and $q^T \geq 0$. Then, for any $N > 0$ there exists a control law $u_k(.)$ that, starting from any initial state $\theta = [\theta_k : k \in \mathcal{K}]$ (I), regulates the network according to the dynamics (D), satisfying conditions (C) and optimizing the payoff function (P).*

It turns out that obtaining the control law amounts to solving a multiparametric linear program. In the below, we assume $N = 1$ and we call the resultant 1-step control law the ORAR control.

The subsequent results state useful properties of the optimal control law: continuity (see Lemma 3.6), stability (see Lemma 3.7), convexity of the value function (see Lemma 3.8), and feasibility (see Lemma 3.9), see the full context in [157, 158, 177, 178].

Thesis 2.3. *The ORAR control law $u(\theta)$ is a continuous and piecewise affine function of θ: $u(\theta) = F_i\theta + g_i$ whenever $\theta \in \mathcal{R}_i$, for some closed polyhedral control regions $\mathcal{R}_i$ in $\mathbb{R}^K$.*

Thesis 2.4. *The ORAR control law $u(.)$ is asymptotically stable.*

Thesis 2.5. *The ORAR control law $u(.)$ optimizes any linear payoff whenever $q_f^T > 0$, $r^T \geq 0$ and $q^T \geq 0$, and the value function is continuous, convex, and piecewise affine.*

Thesis 2.6. *The set of initial states for which the ORAR controller converges in 1 step to the origin (the 1-step feasible set) equals $T(G_c)$.*

The final result gives a new complexity characterization for the ORAR control law. For a full exposition of the result see Theorem 3.10, also [158].

Thesis 2.7. *Given a regular network configuration G_c, consider the ORAR model with an empty payoff, dynamics (D), initial condition (I), constraints (C), and the terminal condition $x_k(1) = 0 : \forall k \in \mathcal{K}$ (T). Then, the number of control regions in the 1-step ORAR control law for the system (D)-(I)-(C)-(T) is upper bounded by the size of the minimal boundary-triangulation of $T(G_c)$.*

6.3 Scalable Internet Routing: The Algebraic Perspective

The next set of results casts the fundamental scalability properties of different routing policies in an unorthodox abstract *algebraic* framework. Consult Table 4.1 for the notation used below.

Thesis 3. *I introduced an algebraic compact routing framework, an extension of compact routing from shortest-path routing to arbitrary routing polices that admit an algebraic definition. I identified the algebraic requirements for a policy to be implementable with sublinear memory at each node (compressible), I showed that certain algebraic properties guarantee superlinear scaling (incompressible), and I gave a comprehensive scalability characterization for most intra-domain routing policies relevant to practice. By generalizing the notion of stretch, I showed that incompressible regular routing policies admit a compressible stretch-3 implementation and I gave the first negative result indicating that certain routing policies remain incompressible even for arbitrary constant stretch.*

The *algebraic compact routing framework* framework rests on a set of definitions (see Section 4.2); namely, routing algebras that give an abstract description of a routing policy (see Definition 4.1), a set of simple algebraic properties of routing algebras, including *monotonicity* and *strict monotonicity*, *isotonicity*, and *regularity* (see Definition 4.2), and *selectivity* (see Definition 4.3), and finally the formal definition of the memory requirements of a routing policy/routing algebra (see Definition 4.5) and the related notions of *compressibility* and *incompressibility*. Our framework characterizes the scalability of individual routing policies in terms of the abstract properties of respective routing algebras.

The first result gives the "good news": if a routing algebra is selective then it scales well. The full result is stated in Theorem 4.9, see also [88, 175, 176, 236].

Thesis 3.1. *If a routing algebra is selective and monotone then it is compressible.*

The second result gives the "bad news": most practical routing algebras, including shortest-path routing, do not admit a scalable implementation. The result and the proof are given in Theorem 4.11, see also [88, 175, 176, 236].

Thesis 3.2. *If a routing algebra is delimited and strictly monotone then it is incompressible.*

Interestingly, just these two claims (with some prior results from the literature, see e.g., Proposition 4.7) are enough to give a comprehensive scalability characterization for most routing policies used in intra-domain Internet routing; see the Table 4.2. With the exception of the shortest-widest-path routing policy, all characterizations are tight.

The following two claims extend the framework from an "optimal routing" context, where the routing algorithm must precisely reproduce the path as requested by the routing policy, to the framework or compact routing, where the algorithm is allowed to digress from the optimal path as long as the weight of the selected path is bounded by a constant *stretch* factor. Here, a routing scheme is said to be of stretch k over algebra $\mathcal{A}$, if for any path p_{st} selected by the scheme: $w(p_{st}) \preceq w(p^*_{st})^k$, where p^*_{st} is a preferred $s-t$ path in $\mathcal{A}$ (see Definition 4.13.

The next result on algebraic compact routing gives the "good news"; see Theorem 4.14 for the full result and see also [88, 175, 176, 236].

Thesis 3.3. *If a routing algebra is delimited and regular then it admits a compressible stretch-3 routing scheme.*

The final claim, however, states a strong negative result: namely, there are routing policies (e.g., the notorious shortest-widest-path policy) that do not admit *any* constant-stretch routing scheme; see Theorem 4.17 and also [88, 175, 176, 236].

Thesis 3.4. *Let $k \geq 1$ and let $\mathcal{A} = (W, \phi, \oplus, \preceq)$ be a monotone algebra with the property that for any $p \geq 2$ there exists a set of weights $\{w_1, w_2, \ldots, w_p\} \subseteq W$ so that $\forall i, j \in \{1, \ldots, p\}, i \neq j$: $w_i \oplus w_j \succ w_i^{2k}$ and $w_i \oplus w_j \succ w_j^{2k}$. Then, there is no stretch-k routing scheme for $\mathcal{A}$ with sublinear memory requirement at all nodes.*

6.4 Scalable Internet Routing: The Information-theoretic Perspective

The final set of results augment the worst-case characterizations from algebraic compact routing to characterize the practical memory requirements of hop-by-hop destination-based routing on *particular* inputs (instead of worst-case graphs). As such, in these set of

results the input graph (or at least some characteristic statistics on it) is also part of the input, contrary to the previous characterizations that hold for *all* graphs. The approach is again interdisciplinary: casting the problem in an *information-theoretic* model certain entropy-like measures are introduced and then the compressed size of forwarding state is described in terms of these entropy notions. Consult Table 5.1 for the notations used below.

The main result is as follows.

Thesis 4. *I presented a series of increasingly tighter entropy measures to characterize the attainable forwarding table compression ratio over an unstructured address space as gradually more and more information about the input forwarding table is revealed to the encoder. I also gave the respective compression schemes that attain these entropy measures, while still providing fast in-place lookup on the compressed state. I considered an idealistic model of the hierarchical Internet routing architecture, I gave an entropy notion for this model, and I designed a forwarding table compression algorithm that attains this entropy bound with fast in-place lookup.*

The "flat" address space model is formally defined in Section 5.3.1. The first result gives tight memory bounds for the *graph independent case*, where we have zero prior knowledge on the problem apart from the network size. The result itself is given in its full version in Theorem 5.1; see also [123, 124].

Thesis 4.1. *Given a graph $G(V, E)$ on n nodes, suppose that the only available information to the encoder is the size n of G. Then, for any $v \in V$ storing the routing function s_v requires at least $n \log n$ bits of space. In addition, there is an encoding scheme that stores s_v in at most $n \log n$ bits and supports lookups in $O(1)$ time.*

The next result strengthens the previous one to the *name-independent* model, where we know the routing policy and hence the exact next-hop port associated with each destination node in the routing function s_v, but we have no control over the assignment of node ids. In this model, we must be able to store the routing function arising over *any* permutation of node ids, even adversarial ones. The corresponding memory bounds are as follows; see Theorem 5.2 and also [123, 124].

Thesis 4.2. *Given a graph $G(V, E)$ on n nodes, suppose that the encoder is aware of the next-hop labels in $s_v : v \in V$ and an adversarial assignment of node ids. Then, storing the routing function s_v for any $v \in V$ requires at least $nH_0(v)$ bits, where $H_0(v) = \sum_{i \in [1, \delta_v]} \frac{n_i}{n} \log \frac{n}{n_i}$ is the empirical zero-order entropy of s_v taken over an arbitrary node id assignment 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. In addition, there is an encoding scheme that stores s_v in at most $nH_0(v) + o(n)$ bits, supporting random access in $O(\log n)$.*

The final characterization for flat address spaces considers the strongest model, the *name-dependent* model. Here, we have control over the assignment of node ids and hence the succession of next-hop symbols in the routing function is known a priori. The space-time characterization in this model is stated at full in Theorem 5.3, see also [123, 124].

Thesis 4.3. *Given a graph G on n nodes, with each node assigned a fixed id in $[1, n]$. Then, encoding the routing function $s_v : v \in V$ for $v \in V$ needs at least $nH_k(v)$ bits*

for any $k > 0$ integer, where $H_k(v) = \sum_{q\in[1,\delta_v]^k} \sum_{i\in[1,\delta_v]} \frac{n_{qi}}{n} \log \frac{n_i}{n_{qi}}$, 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. In addition, there is an encoding scheme that stores s_v in at most $nH_k(v) + o(n)$ bits for any $k = o(\frac{\log n}{\log\log n})$ and supports lookups in $O(\log n \log\log n)$ time.

The next set of results casts the problem in a hierarchical address space that is a better model for the current Internet. In this model, forwarding tables are represented as prefix trees with considerable internal "structure". As such, it is much more difficult to develop the corresponding information-theoretical arguments, since this time the intrinsic structure of the forwarding tables must be correctly reflected by our entropy notions and encoding schemes. The formal model is given in Section 5.4.1.

The main result concerning hierarchical routing gives a tight characterization of the forwarding table size attainable within the hierarchical routing model; the full result is in Theorem 5.4, see also [120, 174, 180, 181].

Thesis 4.4. *Let an IP forwarding table T be given as a proper, binary, leaf-labeled trie T with n leaves and label map s defined on the label alphabet $\Sigma = [1,\delta]$. Then, storing T requires at least $2n + nH_0(s)$ bits of memory, where $H_0(s)$ is the zero-order empirical entropy of the label map s. In addition, there is an encoding scheme that stores T in at most $2n + nH_0(s) + o(n)$ bits so that longest-prefix matching on the compressed representation terminates in $O(W)$ time if $\delta = O(\mathrm{polylog}\, n)$, where W denotes the address width in bits.*

The final result concludes the information-theoretic developments: namely, it can be shown that the higher-order space characterization given under the *flat* model and the zero-order space characterization given under the *hierarchical* model are not independent, but rather they are intricately related to each other. In particular, the following result makes the relation between the size of the representation over a hierarchical address space, M_T, and that of the equivalent flat address space, denoted by M_S, explicit. The full version is in Theorem 5.8, see also [123, 124].

Thesis 4.5. *Given an IP forwarding table, let T be the proper, leaf-labeled binary prefix tree representation and let M_T be the minimum space bound for storing T, and 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. Let M_S be the compressed size of S. Then, $M_T \leq M_S \leq KM_T$ for some $1 \leq K \leq 1 + \frac{1}{2}\log\frac{2^W}{n}$.*

For a typical real IPv4 forwarding table we get $K \sim 3$, for IPv6 $K \sim 7$. The empirical results obtained on real-life IPv4 forwarding tables from [180] (see Table 5.2) indicate that the real value of K might be closer to 1.1–1.5 for IPv4, which gives a very close correspondence between the two metrics.

Bibliography

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

[2] Rachit Agarwal, Anurag Khandelwal, and Ion Stoica. Succinct: Enabling queries on compressed data. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 337–350, Oakland, CA, May 2015. USENIX Association.

[3] S. Agarwal, M. Kodialam, and T. V. Lakshman. Traffic engineering in software defined networks. In *IEEE INFOCOM*, pages 2211–2219, April 2013.

[4] A. K. Agrawala and R. M. Bryant. Models of memory scheduling. In *SOSP*, pages 217–222, 1975.

[5] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, and S. K. Tripathi. Intra-domain QoS routing in IP networks: A feasibility and cost/benefit analysis. *IEEE Network*, 13:42–54, 1999.

[6] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi. Quality of service based routing: A performance perspective. In *ACM SIGCOMM*, pages 17–28, 1998.

[7] D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *ACM SIGCOMM*, pages 313–324, 2003.

[8] 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.

[9] Anthony B Atkinson. On the measurement of inequality. *Journal of Economic Theory*, 2(3):244 – 263, 1970.

[10] D. Awduche. MPLS and traffic engineering in IP networks. *IEEE Communications Magazine*, 37(12):42–47, Dec 1999.

[11] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet traffic engineering. RFC 3272, May 2002.

[12] B. Awerbuch and Y. Shavitt. Topology aggregation for directed graphs. *IEEE/ACM Trans. Netw.*, 9:82–90, February 2001.

[13] Jens Axboe. Linux block IO – present and future. In *Ottawa Linux Symp*, pages 51–61, 2004.

[14] Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Räcke. Optimal oblivious routing in polynomial time. In *ACM symposium on Theory of computing*, STOC '03, pages 383–388, 2003.

[15] Masanori Bando, Yi-Li Lin, and H. Jonathan Chao. FlashTrie: Beyond 100-Gb/s IP route lookup using hash-based prefix-compressed trie. *IEEE/ACM Trans. Netw.*, 20(4):1262–1275, 2012.

[16] 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.

[17] Yair Bartal and Stefano Leonardi. On-line routing in all-optical networks. In *International Colloquium on Automata, Languages and Programming*, ICALP '97, pages 516–526, 1997.

[18] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows*. John Wiley & Sons, New York, second edition, 1990.

[19] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, New York, second edition, 1993.

[20] Alexander Below. *Complexity of triangulation.* Doctoral thesis, Diss., Technische Wissenschaften ETH Zurich, Nr. 14672, 2002, 2002.

[21] A. Bemporad, F. Borrelli, and M. Morari. Explicit solution of LP-based model predictive control. In *IEEE Conference on Decision and Control*, December 2000.

[22] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming – the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, 2002.

[23] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, January 2002.

[24] J. C. R. Bennett and Hui Zhang. WF2Q: Worst-case fair weighted fair queueing. In *IEEE INFOCOM*, volume 1, pages 120–128, 1996.

[25] D. P. Bertsekas. Dynamic behavior of shortest path routing algorithms for communication networks. *IEEE Trans. on Automatic Control*, 27:60–74, 1982.

[26] D. P. Bertsekas and Robert Gallager. *Data Networks.* Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

[27] Jeff Bonwick and Bill Moore. ZFS - the last word in file systems. Sun Microsystems, 2004.

[28] F. Borelli. *Constrained Optimal Control of Linear and Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences.* Springer, 2003.

[29] L. Boróczki. Átvihető folyamokat leíró poliéder keresése hálózatokban (in Hungarian), 2006. Student's Tech. Rep.

[30] F. Borrelli, A. Bemporad, and M. Morari. Geometric algorithm for multiparametric linear programming. *Journal of Optimization Theory and Applications*, 118:515–540, September 2003.

[31] Randal E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[32] S. Büttcher, C.L.A. Clarke, and G.V. Cormack. *Information Retrieval: Implementing and Evaluating Search Engines.* Information Retrieval. MIT Press, 2016.

[33] M. Caesar and J. Rexford. BGP routing policies in ISP networks. Technical Report UCB/CSD-05-1377, EECS Department, University of California, Berkeley, 2005.

[34] Matthew Caesar, Tyson Condie, Jayanthkumar Kannan, Karthik Lakshminarayanan, Ion Stoica, and Scott Shenker. ROFL: Routing on flat labels. *ACM SIGCOMM Computer Communication Review*, 36(4):363–374, 2006.

[35] D. G. Cantor and M. Gerla. Optimal routing in a packet-switched computer network. *IEEE Transactions on Computer*, 23(10):1062–1069, 1974.

[36] Z. Cao and E. W. Zegura. Utility max-min: An application-oriented bandwidth allocation scheme. In *IEEE INFOCOM*, volume 2, pages 793–801, March 1999.

[37] Bogdan Caprita, Wong Chun Chan, Jason Nieh, Clifford Stein, and Haoqiang Zheng. Group ratio round-robin: O(1) proportional share scheduling for uniprocessor and multiprocessor systems. In *USENIX Annual Technical Conference*, pages 337–352, 2005.

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

[39] C.-K. Chau, R. Gibbens, and T. G. Griffin. Towards a unified theory of policy-based routing. In *IEEE INFOCOM*, pages 1–12, 2006.

[40] G. Cheung and S. McCanne. Optimal routing table design for IP address lookups under memory constraints. In *IEEE INFOCOM*, pages 1437–1444, 1999.

[41] Mung Chiang, S.H. Low, A.R. Calderbank, and J.C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

[42] Marco Chiesa, Gabor Retvari, and Michael Schapira. Oblivious routing in IP networks. *IEEE/ACM Transactions on Networking*, 26(3):1292–1305, June 2018.

[43] B. Chinoy and H. W. Braun. The national science foundation network. Tech. Rep., CAIDA, available online: `http://www.caida.org/outreach/papers/1992/nsfn/nsfnet-t1-technology.pdf`, Sep 1992.

[44] Jaeyoung Choi, Jong Han Park, Pei chun Cheng, Dorian Kim, and Lixia Zhang. Understanding BGP next-hop diversity. In *IEEE INFOCOM Workshops*, pages 846 –851, 2011.

[45] A. H. Clifford and G. B. Preston. *The Algebraic Theory of Semigroups, Volume I.* Number 7 in Mathematical Surveys. American Mathematical Society, 1961.

[46] John Cocke. Global common subexpression elimination. *SIGPLAN Not.*, 5(7):20–24, 1970.

[47] ATM Forum Technical Committee. Traffic Management Specification - Version 4.0. ATM Forum/95-0013R13, Feb 1996.

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

[49] L. Cowen. Compact routing with minimum stretch. In *ACM-SIAM SODA'99*, pages 255–260, 1999.

[50] M. Csernai, A. Gulyás, G. Rétvári, Z. Heszberger, and A. Császár. The skeleton of the internet. In *IEEE GLOBECOM*, pages 1–5, November 2010.

[51] L. Csikor, D.M. Divakaran, M.S. Kang, A. Kőrösi, B. Sonkoly, D. Haja, D. Pezaros, S. Schmid, and G. Rétvári. Tuple space explosion: A denial-of-service attack against a software packet classifier. In *ACM CoNEXT*, pages 292–304, 2019.

[52] E. Danna, A. Hassidim, H. Kaplan, A. Kumar, Y. Mansour, D. Raz, and M. Segalov. Upward max min fairness. In *IEEE INFOCOM*, pages 837–845, 2012.

[53] Emilie Danna, Avinatan Hassidim, Haim Kaplan, Alok Kumar, Yishay Mansour, Danny Raz, and Michal Segalov. Upward max-min fairness. *J. ACM*, 64(1):1–24, 2017.

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

[55] Edleno de Moura, Gonzalo Navarro, Nivio Ziviani, and Ricardo Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Trans. Inf. Syst.*, 18(2):113–139, 2000.

[56] Mikael Degermark, Andrej Brodnik, Svante Carlsson, and Stephen Pink. Small forwarding tables for fast routing lookups. In *ACM SIGCOMM*, pages 3–14, 1997.

[57] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM*, pages 1–12, 1989.

[58] R. Denda. The fairness challenge in computer networks. Technical Report TR-00-006, Department for Mathematics and Computer Science, University of Mannheim, 2000.

[59] Sarang Dharmapurikar, Praveen Krishnamurthy, and David E. Taylor. Longest prefix matching using Bloom filters. In *ACM SIGCOMM*, pages 201–212, 2003.

[60] Richard Draves, Christopher King, Srinivasan Venkatachary, and Brian Zill. Constructing optimal IP routing tables. In *IEEE INFOCOM*, 1999.

[61] Will Eatherton, George Varghese, and Zubin Dittia. Tree bitmap: Hardware/software IP lookups with incremental updates. *SIGCOMM Comput. Commun. Rev.*, 34(2):97–122, 2004.

[62] Matthias Englert and Harald Räcke. Oblivious Routing for the L_p-norm. *IEEE Foundations of Computer Science*, pages 32–40, 2009.

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

[64] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *IEEE FOCS*, pages 390–398, 2000.

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

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

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

[68] Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, and Gonzalo Navarro. Compressed representations of sequences and full-text indexes. *ACM Trans. Algorithms*, 3(2), 2007.

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

[70] Simon Fischer, Nils Kammenhuber, and Anja Feldmann. REPLEX: Dynamic traffic engineering based on wardrop routing policies. In *CoNEXT'06*, pages 1–12, 2006.

[71] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, 40(10):118–124, Oct 2002.

[72] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal of Selected Areas in Communications*, 20(4):756–767, May 2002.

[73] P. Fraigniaud and C. Gavoille. Memory requirement for universal routing schemes. In *ACM Symposium on Principles of Distributed Computing*, PODC '95, pages 223–230, 1995.

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

[75] G.N. Frederickson and R. Janardan. Designing networks with compact routing tables. *Algorithmica*, 3(1):171–190, 1988.

[76] Jing Fu, Olof Hagsand, and Gunnar Karlsson. Improving and analyzing LC-trie performance for IP address lookup. *Journal of Networks*, pages 18–27, 2007.

[77] R. Gallager. A minimum delay routing algorithm using distributed computation. *Communications, IEEE Transactions on*, 25(1):73–85, jan 1977.

[78] C. Gavoille. Routing in distributed networks: Overview and open problems. *ACM SIGACT News*, 32(1):52, 2001.

[79] C. Gavoille. An overview on compact routing. In *Workshop on Peer-to-Peer, Routing in Complex Graphs, and Network Coding*, 2007.

[80] C. Gavoille and S. Pérennès. Memory requirement for routing in distributed networks. In *ACM Symposium on Principles of Distributed Computing*, PODC '96, pages 125–133, 1996.

[81] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *USENIX NSDI*, pages 323–336, 2011.

[82] Ali Ghodsi, Matei Zaharia, Scott Shenker, and Ion Stoica. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *ACM EuroSys*, pages 365–378, 2013.

[83] Michel Gondran and Michel Minoux. *Graphs, Dioids and Semirings: New Models and Algorithms*. Operations Research/Computer Science Interfaces Series. Springer Publishing Company, New York, 1 edition, 2008.

[84] P. Grieder, M. Kvasnica, M. Baotic, and M. Morari. Low complexity control of piecewise affine systems with stability guarantee. In *American Control Conference*, volume 2, June 2004.

[85] P. Grieder and M. Morari. Complexity reduction of receding horizon control. *IEEE Conference on Decision and Control*, 3, December 2003.

[86] T. Griffin and J. Sobrinho. Metarouting. In *ACM SIGCOMM*, pages 1–12, 2005.

[87] B. Grünbaum. *Convex Polytopes*. Interscience Publishers John Wiley & Sons, Inc., New York, 1967.

[88] András Gulyás, Gábor Rétvári, Zalán Heszberger, and Rachit Agarwal. On the scalability of routing with policies. *IEEE/ACM Transactions on Networking*, 23(5):1610–1618, October 2015.

[89] S. Gunnar, M. Johansson, and T. Telkamp. Traffic matrix estimation on a large IP backbone: A comparison on real data. In *ACM SIGCOMM Internet Measurement Confernece*, IMC'04, pages 149–160, 2004.

[90] Pankaj Gupta, Balaji Prabhakar, and Stephen P. Boyd. Near optimal routing lookups with bounded worst case performance. In *IEEE INFOCOM*, pages 1184–1192, 2000.

[91] A. Gurney and T. Griffin. Lexicographic products in metarouting. In *Network Protocols, IEEE International Conference on*, pages 113–122, 2007.

[92] Ellen L. Hahne. Round-robin scheduling for max-min fairness in data networks. *IEEE Journal on Selected Areas of Communication*, 9(7):1024–1039, September 1991.

[93] J. Halpern and C. Pignataro. Service function chaining (SFC) architecture, October 2015. RFC 7665.

[94] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley. Multi-Path TCP: A joint congestion control and routing scheme to exploit path diversity in the Internet. *Networking, IEEE/ACM Transactions on*, 14(6):1260 –1271, dec 2006.

[95] Sangjin Han, Keon Jang, KyoungSoo Park, and Sue Moon. PacketShader: a GPU-accelerated software router. In *ACM SIGCOMM*, pages 195–206, 2010.

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

[97] Jahangir Hasan and T. N. Vijaykumar. Dynamic pipelining: Making IP-lookup truly scalable. In *ACM SIGCOMM*, pages 205–216, 2005.

[98] J. He, M. Bresler, M. Chiang, and J. Rexford. Towards robust multi-layer traffic engineering: Optimization of congestion control and routing. *Selected Areas in Communications, IEEE Journal on*, 25(5):868–880, June 2007.

[99] Jiayue He, Martin Suchara, Ma'ayan Bresler, Jennifer Rexford, and Mung Chiang. Rethinking Internet traffic management: From multiple decompositions to a practical protocol. In *ACM CoNEXT'07*, pages 1–12, 2007.

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

[101] G. Huston. BGP routing table analysis reports. http://bgp.potaroo.net/.

[102] G. Huston. Interconnection, peering, and settlements. In *INET*, 1999.

[103] Internet Live Stats. `https://www.internetlivestats.com`.

[104] Ioannis Ioannidis and Ananth Grama. Level compressed DAGs for lookup tables. *Comput. Netw.*, 49(2):147–160, 2005.

[105] M. Iri. On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13(3):129–135, 1971.

[106] G. Jacobson. Space-efficient static trees and graphs. In *IEEE FOCS*, pages 549–554, 1989.

[107] J. M. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, 29(7):954–962, July 1981.

[108] Raj Jain, Dah-Ming Chiu, and William R Hawe. *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*, volume 38. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984.

[109] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang. Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework. In *IEEE INFOCOM*, pages 1206–1214, 2012.

[110] Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. Multiresource allocation: Fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM Transactions on Networking*, 21(6):1785–1798, 2013.

[111] Rawls John. Justice as fairness: A restatement. *Harvard University Press, Cambridge, Massachusetts*, 2001.

[112] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In *ACM SIGCOMM*, August 2005.

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

[114] J. Katajainen and E. Mäkinen. Tree compression and optimization with applications. *International Journal of Foundations of Computer Science*, 1(4):425–447, 1990.

[115] F P Kelly, A K Maulloo, and D K H Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, Mar 1998.

[116] Frank Kelly and Thomas Voice. Stability of end-to-end algorithms for joint routing and rate control. *SIGCOMM Comput. Commun. Rev.*, 35(2):5–12, 2005.

[117] P. Key, L. Massoulie, and P.D. Towsley. Path selection and multipath congestion control. In *IEEE INFOCOM*, pages 143 –151, May 2007.

[118] A. Khanna and J. Zinky. The revised ARPANET routing metric. *SIGCOMM Comput. Commun. Rev.*, 19(4):45–56, 1989.

[119] 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.

[120] 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 *Proc. IEEE ICNP*, 2014.

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

[122] Murali Kodialam, T. V. Lakshman, and Sudipta Sengupta. Efficient and robust routing of highly variable traffic. In *ACM Hot Topics in Networks (HotNets)*, 2004.

[123] A. Kőrösi, A. Gulyás, Z. Heszberger, J. Bíró, and G. Rétvári. On the memory requirement of hop-by-hop routing: Tight bounds and optimal address spaces. *IEEE/ACM Transactions on Networking*, pages 1–11, 2020.

[124] A. Kőrösi and G. Rétvári. A csomagtovábbítás skálázhatósága: korlátok és optimumok. *ALKALMAZOTT MATEMATIKAI LAPOK*, 36, 2019.

[125] D. Krioukov, K. Fall, and X. Yang. Compact routing on Internet-like graphs. In *IEEE INFOCOM*, volume 1, 2004.

[126] D. Krioukov, kc claffy, K. Fall, and A. Brady. On compact routing for the Internet. *ACM Comp. Comm. Review*, 37(3):41–52, 2007.

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

[128] Thomas S Kuhn. *The structure of scientific revolutions*. University of Chicago press, 2012.

[129] A. Kvalbein, C. Dovrolis, and C. Muthu. Multipath load-adaptive routing: putting the emphasis on robustness and simplicity. In *IEEE International Conference on Network Protocols*, ICNP 2009, pages 203–212, oct. 2009.

[130] Constantino M. Lagoa, Hao Che, and Bernardo A. Movsichoff. Adaptive control algorithms for decentralized optimal traffic engineering in the Internet. *IEEE/ACM Trans. Netw.*, 12(3):415–428, 2004.

[131] Tian Lan and Mung Chiang. An axiomatic theory of fairness in resource allocation. available online: `http://www.princeton.edu/~chiangm/fairness.pdf`, 2012.

[132] J. Le Boudec. Rate adaptation, congestion control and fairness: A tutorial. available online: `http://ica1www.epfl.ch/PS_files/LEB3132.pdf`, Feb 2005.

[133] J. Le Boudec and B. Radunovic. A unified framework for max-min and min-max fairness with applications. In *Annual Allerton Conference on Communication, Control, and Computing*, Oct 2002.

[134] Jang-Won Lee, Ravi R. Mazumdar, and Ness B. Shroff. Non-convex optimization and rate control for multi-class services in the Internet. *IEEE/ACM Trans. Netw.*, 13(4):827–840, 2005.

[135] W. Lee, M. Hluchyi, and P. Humblet. Routing subject to quality of service constraints in integrated communication networks. *IEEE Network Magazine*, 9(4):46–55, July-August 1999.

[136] T. Lévai, G. Pongrácz, P. Megyesi, P. Vörös, S. Laki, F. Németh, and G. Rétvári. The price for programmability in the software data plane: The vendor perspective. *IEEE Journal on Selected Areas in Communications*, 36(12):2621–2630, December 2018.

[137] Yuxi Li, Baochun Bai, Janelle J. Harms, and Robert Holte. Stable and robust multipath oblivious routing for traffic engineering. In *International Teletraffic Congress*, volume 4516 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 2007.

[138] Yaoqing Liu, Syed Obaid Amin, and Lan Wang. Efficient FIB caching using minimal non-overlapping prefixes. *SIGCOMM Comput. Commun. Rev.*, 43(1):14–21, January 2012.

[139] LuceneTransform. Transparent compression for Apache Lucene. `http://code.google.com/p/lucenetransform`.

[140] Qingming Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. In *International Conference on Network Protocols (ICNP '97)*, page 191, 1997.

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

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

[143] A.J. McAuley and P. Francis. Fast routing table lookup using CAMs. In *IEEE INFOCOM*, pages 1382–1391, 1993.

[144] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

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

[146] D. Medhi. Multi-hour, multi-traffic class network design for virtual path-based dynamically reconfigurable wide-area ATM networks. *IEEE/ACM Transactions on Networking*, 3(6):809–818, 1995.

[147] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *ACM SIGCOMM*, pages 161–174, 2002.

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

[149] Jun-Ki Min, Myung-Jae Park, and Chin-Wan Chung. XPRESS: a queriable compression for XML data. In *ACM SIGMOD*, pages 122–133, 2003.

[150] D. Nace and L.N Doan. A polynomial approach to the fair multi-flow problem. Tech. Rep., Heudiasyc, UTC, available online: `http://www.hds.utc.fr/~dnace/recherche/Publication/TR-MMF.pdf`, 2002.

[151] M. Nagy, J. Tapolcai, and G. Rétvári. R3D3: A doubly opportunistic data structure for compressing and indexing massive data. *Infocommunications Journal*, 11(2):58–66, January 2019.

[152] Máté Nagy, János Tápolcai, and Gábor Rétvári. Node virtualization for IP level resilience. *IEEE/ACM Transactions of Networking*, 26(3):1250–1263, June 2018.

[153] John F Nash Jr. The bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 155–162, 1950.

[154] Gonzalo Navarro and Francisco Claude. libcds: Compact data structures library, 2004. `http://libcds.recoded.cl`.

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

[156] Gonzalo Navarro and Jorma Tarhio. LZgrep: a Boyer-Moore string matching tool for Ziv-Lempel compressed text. *Softw. Pract. Exper.*, 35(12):1107–1130, 2005.

[157] G. Németh and G. Rétvári. Hybrid demand oblivious routing: Hyper-cubic partitions and theoretical upper bounds. In *7th International ICST Conference on Broadband Communications, Networks, and Systems (BROADNETS 2010)*, pages 25–27, 2010.

[158] G. Németh and G. Rétvári. Rate-adaptive multipath routing: Distributed, centralized, and hybrid architectures. *Networks*, pages 1–12, 2015.

[159] S. Nilsson and G. Karlsson. IP-address lookup using LC-tries. *IEEE JSAC*, 17(6):1083 –1092, 1999.

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

[161] Engineering Committee on Science, Public Policy, Institute of Medicine (US), National Academies (US). Committee on Facilitating Interdisciplinary Research, National Academy of Engineering (US), and National Academy of Sciences (US). *Facilitating interdisciplinary research*. National Academies Press, 2004.

[162] K. Onaga and O. Kakusho. On feasibility conditions of multicommodity flows in networks. *IEEE Transactions on Circuit Theory*, 18(4):425–429, 1971.

[163] Harald Räcke. Minimizing congestion in general networks. In *IEEE Symposium on Foundations of Computer Science*, FOCS '02, pages 43–52, 2002.

[164] Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *ACM symposium on Theory of computing*, STOC '08, pages 255–264, 2008.

[165] Harald Räcke. Survey on oblivious routing strategies. In *Conference on Computability in Europe: Mathematical Theory and Computational Practice*, CiE '09, pages 419–429, 2009.

[166] Rajeev Raman, Venkatesh Raman, and S. Srinivasa Rao. Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. In *ACM-SIAM SODA*, pages 233–242, 2002.

[167] J. Rawlings and K. Muske. The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 38(10):1512–1516, 1993.

[168] 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.

[169] G. Rétvári. The geometry of networking, part I. In *Proceedings of High Speed Networking 2007 Spring Workshop*, Balatonkenese, Hungary, May 2007.

[170] G. Rétvári. The geometry of networking, part II. In *Proceedings of High Speed Networking 2009 Spring Workshop*, Balatonkenese, Hungary, May 2009.

[171] G. Rétvári, J. J. Bíró, and T. Cinkler. Fairness in capacitated networks: a polyhedral approach. In *IEEE INFOCOM 2007*, Anchorage, Alaska, USA, May 2007.

[172] G. Rétvári, J. J. Bíró, and T. Cinkler. On shortest path representation. *IEEE/ACM Transactions on Networking*, 15(6):1293–1306, December 2007.

[173] G. Rétvári, J. J. Bíró, and T. Cinkler. Routing-independent fairness in capacitated networks. In *Proc., IEEE International Conference on Communications (ICC 2007)*, Glasgow, Scotland, June 2007.

[174] G. Rétvári, Z. Csernátony, A. Kőrösi, J. Tapolcai, A. Császár, G. Enyedi, and G. Pongrácz. Compressing IP forwarding tables for fun and profit. In *ACM HotNets-XI*. ACM, October 2012.

[175] G. Rétvári, A. Gulyás, Z. Heszberger, M. Csernai, and J.J. Bíró. Compact policy routing. In *ACM PODC 2011*, pages 149–158, New York, NY, USA, 2011. ACM.

[176] G. Rétvári, A. Gulyás, Z. Heszberger, M. Csernai, and J.J. Bíró. Compact policy routing. *Distributed Computing*, 26(5):309–320, 2013.

[177] G. Rétvári and G. Németh. Demand-oblivious routing: Distributed vs. centralized approaches. In *IEEE INFOCOM 2010*, March 2010.

[178] G. Rétvári and G. Németh. On optimal multipath rate-adaptive routing. In *15th IEEE Symposium on Computers and Communications (ISCC 2010)*, Riccione, Italy, June 2010.

[179] G. Rétvári, D. Szabó, A. Gulyás, A. Kőrösi, and J. Tapolcai. An information-theoretic approach to routing scalability. In *ACM HotNets-XIII*, pages 2:1–2:7, 2014.

[180] G. Rétvári, J. Tapolcai, A. Kőrösi, A. Majdán, and Z. Heszberger. Compressing IP forwarding tables: Towards entropy bounds and beyond. In *ACM SIGCOMM*, pages 111–122, 2013.

[181] G. Rétvári, J. Tapolcai, A. Kőrösi, A. Majdán, and Z. Heszberger. Compressing IP forwarding tables: Towards entropy bounds and beyond. *IEEE/ACM Transactions on Networking*, 24(1):149–162, 2016.

[182] Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Trans. Visual Comput. Graphics*, 5:47–61, 1999.

[183] M. Roughan, M. Thorup, and Y. Zhang. Traffic engineering with estimated traffic matrices. In *ACM SIGCOMM Internet measurement Conference*, IMC '03, pages 248–258, 2003.

[184] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. In *ACM SIGCOMM Workshop on Internet measurment*, IMW '02, pages 91–92, 2002.

[185] Alfréd Rényi. On measures of entropy and information. In *Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 547–561. University of California Press, 1961.

[186] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Algorithms and Combinatorics. Springer, 2002.

[187] A. Seehra, J. Naous, M. Walfish, D. Mazieres, A. Nicolosi, and S. Shenker. A policy framework for the future Internet. *ACM Workshop on Hot Topics in Networks (HotNets)*, 2009.

[188] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[189] M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, 1996.

[190] Keith Sklower. A tree-based packet routing table for Berkeley UNIX. Technical Report, Berkeley, 1991.

[191] J. Sobrinho. Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet. *IEEE/ACM Trans. Netw.*, 10:541–550, August 2002.

[192] J. Sobrinho. Network routing with path vector protocols: Theory and applications. In *ACM SIGCOMM*, pages 49–60, 2003.

[193] Haoyu Song, Murali S. Kodialam, Fang Hao, and T. V. Lakshman. Scalable IP lookups using Shape Graphs. In *IEEE ICNP*, pages 73–82, 2009.

[194] Haoyu Song, Jonathan Turner, and John Lockwood. Shape shifting tries for faster IP route lookup. In *IEEE ICNP*, pages 358–367, 2005.

[195] E. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*, volume 6 of *Texts in Applied Mathematics*. Springer, 1998. available online: `http://www.math.rutgers.edu/~sontag/FTP_DIR/sontag_mathematical_control_theory_springer98.pdf`.

[196] N. Spring, , R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *ACM SIGCOMM*, pages 113–124, 2003.

[197] V. Srinivasan and George Varghese. Faster IP lookups using controlled prefix expansion. *SIGMETRICS Perform. Eval. Rev.*, 26(1):1–10, 1998.

[198] Marilyn Stember. Advancing the social sciences through the interdisciplinary enterprise. *The Social Science Journal*, 28(1):1–14, 1991.

[199] S. Stergiou and J. Jain. Optimizing routing tables on systems-on-chip with content-addressable memories. In *System-on-Chip*, pages 1–6, 2008.

[200] Ion Stoica, Hussein Abdel-Wahab, Kevin Jeffay, Sanjoy K Baruah, Johannes E Gehrke, and C Greg Plaxton. A proportional share resource allocation algorithm for real-time, time-shared systems. In *IEEE Real-Time Systems Symposium*, pages 288–299, 1996.

[201] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks. In *ACM SIGCOMM*, pages 118–130, 1998.

[202] Hong Sun, Ozgur Ozturk, and Hakan Ferhatosmanoglu. CoMRI: A compressed multi-resolution index structure for sequence similarity queries. In *IEEE CSB*, pages 553–, 2003.

[203] S. Suri, M. Waldvogel, and P. R. Warkhede. Profile-based routing: a new framework for MPLS traffic engineering. In F. Boavida, editor, *Quality of Future Internet Services*, volume 2156 of *LNCS*. Springer, 2001.

[204] J. Tapolcai, G. Rétvári, P. Babarczi, and E. R. Bérczi-Kovács. Scalable and efficient multipath routing via redundant trees. *IEEE Journal on Selected Areas in Communications*, 37(5):982–996, May 2019.

[205] Renata Teixeira, Sharad Agarwal, and Jennifer Rexford. BGP routing changes: Merging views from two ISPs. *SIGCOMM Comput. Commun. Rev.*, 35:79–82, October 2005.

[206] Renata Teixeira, Nick G. Duffield, Jennifer Rexford, and Matthew Roughan. Traffic matrix reloaded: Impact of routing changes. In *PAM'05*, pages 251–264, 2005.

[207] Renata Teixeira, Keith Marzullo, Stefan Savage, and Geoffrey M. Voelker. In search of path diversity in ISP networks. In *ACM IMC*, pages 313–318, 2003.

[208] Renata Teixeira, Aman Shaikh, Tim Griffin, and Jennifer Rexford. Dynamics of hot-potato routing in IP networks. In *Conference on Measurement and modeling of computer systems*, SIGMETRICS '04/Performance '04, pages 307–319, 2004.

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

[210] Pankaj M. Tolani and Jayant R. Haritsa. XGRIND: a query-friendly XML compressor. In *ICDE*, pages 225–234, 2002.

[211] Zartash Afzal Uzmi, Markus Nebel, Ahsan Tariq, Sana Jawad, Ruichuan Chen, Aman Shaikh, Jia Wang, and Paul Francis. SMALTA: Practical and near-optimal FIB aggregation. In *ACM CoNEXT*, pages 1–12, 2011.

[212] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *ACM symposium on Theory of computing*, STOC '81, pages 263–277, 1981.

[213] Hal R Varian. Equity, envy, and efficiency. *Journal of economic theory*, 9(1):63–91, 1974.

[214] Sebastiano Vigna and Paolo Boldi. MG4J: Managing Gigabytes for Java. `http://mg4j.dsi.unimi.it`, 2007.

[215] Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner. Scalable high speed IP routing lookups. In *ACM SIGCOMM*, pages 25–36, 1997.

[216] F. Wang and L. Gao. On inferring and characterizing Internet routing policies. In *ACM SIGCOMM Internet Measurement Conference*, pages 15–26, 2003.

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

[218] Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. COPE: traffic engineering in dynamic networks. *SIGCOMM Comput. Commun. Rev.*, 36(4):99–110, 2006.

[219] Zheng Wang and Jon Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal of Selected Areas in Communications*, 14(7):1228–1234, 1996.

[220] WebGraph. A framework for graph compression. `http://webgraph.di.unimi.it`.

[221] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, 1999.

[222] J. Woods. PPP Deflate Protocol. RFC 1979, 1996.

[223] Weidong Wu. *Packet Forwarding Technologies*. Auerbach, 2008.

[224] D. Xu, M. Chiang, and J. Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *Networking, IEEE/ACM Transactions on*, 19(6):1717–1730, 2011.

[225] Kuai Xu, Zhi-Li Zhang, and Supratik Bhattacharyya. Profiling Internet backbone traffic: Behavior models and applications. *SIGCOMM Comput. Commun. Rev.*, 35:169–180, August 2005.

[226] L. Yang, R. Dantu, T. Anderson, and R. Gopal. Forwarding and control element separation (ForCES) framework. RFC 3746, April 2004.

[227] Tong Yang, Gaogang Xie, YanBiao Li, Qiaobin Fu, Alex X. Liu, Qi Li, and Laurent Mathy. Guarantee IP lookup performance with FIB explosion. In *ACM SIGCOMM*, ACM SIGCOMM, page 39–50, 2014.

[228] O. Younis and S. Fahmy. Constraint-based routing in the Internet: Basic principles and recent research. *Communications Surveys Tutorials, IEEE*, 5(1):2–13, 2003.

[229] Jung-Hoon Yun, Anseok Lee, and Song Chong. Multi-path aggregate flow control for real-time traffic engineering. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1 –5, 2008.

[230] Doron Zarchy, David Hay, and Michael Schapira. Capturing resource tradeoffs in fair multi-resource allocation. In *IEEE INFOCOM*, pages 1062–1070, 2015.

[231] Marko Zec, Luigi Rizzo, and Miljenko Mikuc. DXR: Towards a billion routing lookups per second in software. *SIGCOMM Comput. Commun. Rev.*, 42(5):29–36, 2012.

[232] C. Zhang, Y. Liu, W. Gong, J. Moll, and R. D. Towsley. On optimal routing with multiple traffic matrices. In *IEEE INFOCOM*, volume 1, pages 607–618, 2005.

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

[234] G.M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer, New York, 1998.

[235] 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.

[236] Márton Zubor, Attila Korösi, András Gulyás, and Gábor Rétvári. On the computational complexity of policy routing. In Yvon Kermarrec, editor, *Advances in Communication Networking*, volume 8846 of *Lecture Notes in Computer Science*, pages 202–214. Springer International Publishing, 2014.

Index