GMA: A Pareto Optimal Distributed Resource-Allocation Algorithm

Giacomo Giuliari, Marc Wyss, Markus Legner, and Adrian Perrig

ETH Zürich, Universitätstrasse 6, 8092 Zürich, Switzerland
{giacomog, marc.wyss, markus.legner, adrian.perrig}@inf.ethz.ch

Abstract To address the raising demand for strong packet delivery guarantees in networking, we study a novel way to perform graph resource allocation. We first introduce *allocation graphs*, in which nodes can independently set local resource limits based on physical constraints or policy decisions. In this scenario we formalize the *distributed path-allocation* (PA^{dist}) problem, which consists in allocating resources to paths considering only *local* on-path information—importantly, not knowing which other paths could have an allocation—while at the same time achieving the *global* property of never exceeding available resources.
Our core contribution, the global myopic allocation (GMA) algorithm, is a solution to this problem. We prove that GMA can compute *unconditional* allocations for all paths on a graph, while never over-allocating resources. Further, we prove that GMA is Pareto optimal with respect to the allocation size, and it has linear complexity in the input size. Finally, we show with simulations that this theoretical result could be indeed applied to practical scenarios, as the resulting path allocations are large enough to fit the requirements of practically relevant applications.

1 Introduction

Allocating resources such as bandwidth in a network has proven to be a difficult problem from both a theoretical and practical perspective: in many cases, networks consist of independent nodes without central controller and without a global view of the topology and available resources. Furthermore, these nodes often have their own policies on how to allocate resources. To the best of our knowledge, the theoretical networking literature is lacking solutions that address this distributed setting. In this paper, we consider *allocation graphs*, directed graphs consisting of independent nodes augmented with local policies, i.e., the amount of resources each node allocates for transit between any pair of neighbors. While we interpret the resources as bandwidth, other interpretations—like computations on behalf of the neighbors—are possible as well.

For any path in the allocation graph, we want to *myopically* compute a static allocation, i.e., based only on the local policies of on-path nodes. This allocation should guarantee that no local allocation is ever exceeded, even when all path allocations in the network are fully used simultaneously. This is resource allocation is therefore *unconditional*, since the size of one allocation is completely

independent of any other allocation, and not determined by an admission process, and thus cannot be influenced by single off-path nodes. In particular, nodes do not need to keep track of allocations as each individual allocation is valid independently of whether or not any other allocations are used. We formalize the problem of finding the size of such allocations as the *distributed path allocation* (PA^{dist}) problem. Two major questions then arise: (i) Can unconditional resource allocation indeed be performed in a distributed setting, where nodes have only partial information on the network, without creating over-allocation? And (ii), since an allocation is implicitly created for every path in the network, can allocations be large enough to be useful in practice?

Our work addresses these problems, finding that it is possible to both avoid over-allocation and create allocations that meet the demands of a number of modern critical applications at the same time. We show this constructively, by proposing the first unconditional resource allocation algorithm: the *global myopic allocation* (GMA) algorithm. GMA interprets each node's local allocations both as capacity limits that must not be exceeded and as policy decisions about the relative importance of links to neighbors. It efficiently computes allocations that scale with these local policies, and ensures that capacities are not over-allocated. We prove that GMA fulfills all desired properties and that it is *Pareto optimal* with respect to all other algorithms that solve the PA^{dist} problem. Finally, we simulate GMA on random graphs, chosen to model real-world use cases; we evaluate the size of the resulting path allocations and show that they are viable for practical applications.

Practical relevance of the PA^{dist} problem. Over the past decades, computer networks have predominantly relied on the *best-effort* paradigm. Endpoints run congestion-control algorithms to prevent a congestion-induced collapse of the network [10, 12], but no further guarantees for packet delivery or quality of service can be given. This has been shown to work reasonably well for many applications like web browsing, but it is becoming increasingly clear that it is far from optimal in terms of performance and fairness [18, 7].

Although the networking community has developed several protocols to reserve resources for individual connections [15, 4, 3], none of them has seen widespread adoption because of their high complexity and poor scalability. These drawbacks arise in all these systems as they offer *conditional* allocations: endpoints can select the amount of resource to allocate, the rationale being that supply and demand will eventually lead to optimal resource utilization. However, this also means that all nodes have to store information about all individual requests, and check that new requests do not exceed resource capacity.

An unconditional resource allocation system based on the GMA algorithm avoids this problem. In a network of compliant sources using such a system, nodes do not need to keep track of allocations as each allocation is valid independently of whether or not any other allocations are used. Further, GMA guarantees that no over-allocation of bandwidth—and therefore congestion—occurs. Thus, strong delivery guarantees can be provided to the communications in this network, without the overhead required by conditional systems. Appendix A

presents overview of the *critical* applications that would benefit the most from an unconditional resource allocation system.

2 Preliminaries: formalizing resource allocation

We now introduce the formalism we use throughout the paper, and characterize the path-allocation (PA) problem. Although the PA problem arises from an applied networking context (as some of the terminology also suggests), we seek to provide a formulation that is not tied to networking, such that our solution can also be applied to other areas. Therefore, we define the problem with the abstraction of *allocation* graphs.

Allocation graphs. We augment the standard directed graph definition, comprising nodes and edges, with a set of *interfaces* at every node.[1] An interface denotes the end of one of the edges attached to a node, while a *local interface*, which is not associated with any edge, represents internal sources or sinks (these concepts are shown in Figs. 1a and 1b on page 6). In an allocation graph, a *resource*—a generic quantity of interest—is associated with edges, and is a measure of supply. The *capacity* of an edge is a fixed, positive real number that represents the maximum amount of resource it can provide;[2] it is denoted by $cap^{(k)}_{i,\mathrm{IN}}$, for the capacity of the edge incoming to interface i of node k, and $cap^{(k)}_{i,\mathrm{OUT}}$ for the outgoing edge. Further, we assume that an *allocation matrix* $M^{(k)}$ is given for each node k. Allocation matrices are illustrated in Figs. 1b and 1c. An entry $M^{(k)}_{i,j}$ in the allocation matrix, called *pair allocation*, denotes the maximum amount of resource that can be allocated in total to all the paths incoming at interface i and outgoing at interface j. Allocation matrices are non-negative and not necessarily symmetric. We call the maximum amount of resource that can be allocated from an interface i to every other interface the *divergent*, and the maximum amount of resource that can be allocated from every other interface towards an interface j the *convergent*. They are calculated as the sum of rows or columns of $M^{(k)}$, respectively:

$$DIV^{(k)}_i = \sum_j M^{(k)}_{i,j}, \qquad CON^{(k)}_j = \sum_i M^{(k)}_{i,j}. \tag{1}$$

The matrix $M^{(k)}$ must be defined to fulfill $\forall i.\ DIV^{(k)}_i \leq cap^{(k)}_{i,\mathrm{IN}}, CON^{(k)}_i \leq cap^{(k)}_{i,\mathrm{OUT}}$, that is, neither $DIV^{(k)}_i$ nor $CON^{(k)}_i$ respectively exceed the capacity of the incoming and outgoing edges, connected to interface i of node k.

Intuitively, an *interface pair* (i, j) is the logical connection between two interfaces of a node, and thus a pair allocation expresses the maximum amount of resource the node is willing to provide from one neighbor to the next. Allocation

[1] A node can be thought of as, e.g., an autonomous system in the Internet, or any other entity part of a distributed system that acts independently from other entities.

[2] We use dimensionless values for the resource; in practice, these could correspond to, e.g., bandwidth (in Gbps) or computations per second.

matrices can therefore be seen as a way for nodes to encode policies on the level of service they want to grant to each pair of neighbors.

In this model, we represent a path of ℓ nodes $N^1, \ldots, N^\ell$ as a list of nodes and interface pairs $\pi = [(N^1, i^1, j^1), (N^2, i^2, j^2), \ldots, (N^\ell, i^\ell, j^\ell)]$.[3] To simplify the presentation, we will omit the nodes from the list when they are implicitly clear; we will also use the abbreviation $M_{i,j}^{(k)} \equiv M_{i^k,j^k}^{(N^k)}$. We say that a path is *terminated*, if the first interface of the first pair and the second interface of the last pair are local interfaces. Otherwise the path is called *preliminary*. A path is considered *simple* or *loop-free*, if it contains each node at most once. Furthermore, we use π^k to denote the *preliminary prefix-path of length k* of some terminated path π of length ℓ ($\pi^k = [(i^1, j^1), (i^2, j^2), \ldots, (i^k, j^k)]$ for $1 \leq k < \ell$). Finally, we call a path *valid*, if $M_{i,j}^{(1)}, \ldots, M_{i,j}^{(\ell)} > 0$, otherwise it is *invalid*.

The PA problem. We are interested in the problem of allocating the resource on an allocation graph to paths. A *path allocation* is created when a certain amount of resource is allocated for that path, exclusively reserving this amount on every edge and interface pair of the path and thus making it unavailable for any other path. If the sum of the path allocations traversing an edge exceeds the capacity of the edge, we say that the edge is *over-allocated*. Similarly, an interface pair (i^k, j^k) is over-allocated if this sum is larger than its corresponding pair allocation $M_{i,j}^{(k)}$.

Given an allocation graph and information on the allocation matrices, the PA problem is to calculate a path allocation for any path π in this graph with the following constraint:

C1 No-over-allocation: For all allocation graphs, even if there is an allocation on every possible valid path in the graph, no edge and no interface pair is ever over-allocated.[4]

Solving the PA problem then requires finding an algorithm $\mathcal{A}$ that can compute such path allocations. We intentionally left underspecified the precise input that such an algorithm receives, as it depends on whether the algorithm is centralized or distributed. If centralized, $\mathcal{A}$'s input is the whole network topology, as well as the allocation matrices of all the nodes. Thus, the centralized PA problem can be viewed as a variant of the multicommodity flow problem [9], with the additional constraint that pair allocations have to be respected.

In the distributed version of the PA problem (PA^{dist}), the algorithm has to run consistently on each node, with partial information about the allocation graph. Since nodes on a path are assumed to be able to exchange information, we restrict this information by requiring $\mathcal{A}$'s input to contain only information about the path for which the path allocation is computed. This is captured by the following definition:

[3] This definition implicitly includes edges. Also, we assume that the interfaces match, i.e., $j^{(k-1)}$ and $i^{(k)}$ are interfaces at opposite ends of the same directed edge.

[4] Paths with loops, and of arbitrary length, are also included in this definition.

The PA^{dist} problem is to solve the PA problem with this additional restriction:

C2 Locality: The path allocation is a function of the on-path allocation matrices $M^{(1)}, \ldots, M^{(\ell)}$ only.

Among the set of algorithms that fit this definition, we are naturally interested in the ones that lead to higher path allocations. Since a precise optimality condition on the algorithm depends on the practical application for which it is used, we generally postulate that meaningful algorithms provide path allocations that cannot be strictly increased. This is captured by Pareto optimality:

Opt Optimality: Consider the class of all algorithms fulfilling the requirement of either PA or PA^{dist}. Algorithm $\mathcal{A}$ from this class is (Pareto) optimal if there is no other algorithm $\mathcal{B}$ from the same class that can provide at least the same path allocation for every path of every allocation graph, and a strictly better allocation for at least one path. Formally, if there exists a graph with a path π for which $\mathcal{B}(\pi) = \mathcal{A}(\pi) + \delta$ with $\delta > 0$, then there exists at least one other path π', possibly in a different graph, where $\mathcal{B}(\pi') = \mathcal{A}(\pi') - \delta'$ with $\delta' > 0$.[5]

In addition, we specify three supplementary properties that make an algorithm more amenable to practical settings. First, the algorithm should provide non-zero allocations for all valid paths, second, we require the algorithm to be efficient in the length of the path and the size of the on-path allocation matrices, and lastly, we enforce stricter requirements on the policy of individual nodes with the *monotonicity* property: if a node increases one of its pair allocations, we expect all path allocations crossing the interface pair to at least not decrease. Increasing one pair allocation also increases the corresponding divergent and convergent, while all other pair allocations that contribute to this convergent or divergent remain the same. Therefore the relative contribution of the increased pair allocation becomes higher, while the relative contribution of the other pair allocations decreases. This way, a node's allocation matrix can also be understood as a policy that defines the relative importance of its neighbors. Since a path containing loops might traverse the same node both through a pair allocation with increased importance and through one with decreased importance, monotonicity is only meaningful in the context of simple paths. [6]

S1 Usability: For every valid path π, the resulting allocation is positive ($\mathcal{A}(\pi) > 0$).

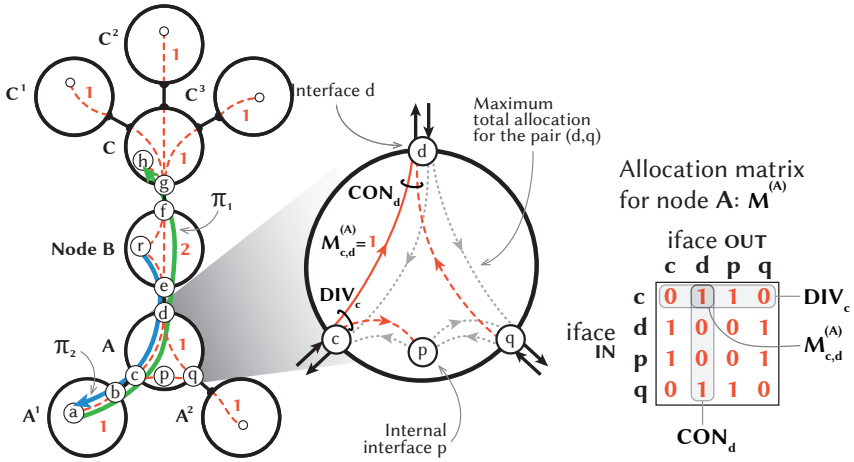
S2 Efficiency: Algorithm $\mathcal{A}$ should have at most polynomial complexity as a function of input size. Specifically, for PA^{dist} this means polynomial in the total size of the allocation matrices of on-path nodes. This is a relatively loose requirement, we will show a linear algorithm in the following.

[5] The loose constraint that π' is possibly in a different graph comes from the fact that because of the locality property in the PA^{dist} problem, the algorithm has no way to differentiate two graphs having a path with the same nodes and allocation matrices.

[6] For $i_1^k \neq i_2^k$, increasing $M^{(k)}_{i_1,j}$ decreases the relative contribution of $M^{(k)}_{i_2,j}$ (Eq. (1)).

S3 Monotonicity: If the pair allocation of some node k on a simple path π is increased and all other allocations remain unchanged, the resulting allocation must not decrease: $M_{i,j}^{(k)} \leq \widetilde{M}_{i,j}^{(k)} \implies \mathcal{A}(\pi) \leq \widetilde{\mathcal{A}}(\pi)$.

The challenge of devising an optimal PA$^{\text{dist}}$ algorithm is clear: $\mathcal{A}$ can only rely on a *myopic* view of the path, without any further knowledge about the larger graph. However, it has to achieve the *global* constraints of Pareto-optimality and no-over-allocation, which consider the result of performing allocations on all valid paths. In the remainder of the paper, we present the global myopic allocation (GMA) algorithm as a solution to the PA$^{\text{dist}}$ problem. GMA fulfills requirements C1 and C2, and is optimal according to Opt, which we formally prove in §4. Furthermore, we prove in Appendix E that GMA also satisfies all the supplementary requirements (S1–S3). An additional property, *extensibility*, is presented and proven in Appendix F.



(a) *An allocation graph.* **(b)** *Detailed structure of* A. **(c)** *Allocation matrix of* A.

Figure 1: ***Example of an allocation graph.*** *Pair allocations are represented in Fig. 1a by dashed lines—their size shown by the number in the respective node. If two interfaces are not connected by dashed lines, their pair allocation is zero. All pair allocations are bidirectional, as shown in Fig. 1b. For clarity, we use globally unique interface identifiers. Figure 1a also shows paths π_1 and π_2, used in the examples (π_3 is the reverse of π_2).*

3 Introducing the GMA algorithm

We present the GMA algorithm in three steps: starting from a simple first-cut approach, at each step we present a refinement of the previous algorithm. This section is meant to provide a profound yet intuitive understanding of the GMA algorithm and its properties—accompanied by the example in Fig. 1a—leading to the final formulation of GMA in Eq. (10).

3.1 Step 1: towards no-over-allocation

As a first attempt to achieve no-over-allocation, we take the pair allocation of the first node on a path, and multiply it by the ratio of the pair allocation and the divergent for each of the traversed interface pairs. With this approach, each node receives a *preliminary allocation* from the previous node, fairly splits it among all interfaces according to the pair allocations, and passes it on to the next node. This leads to the following formula:

$$\mathcal{A}_1(\pi) = M_{i,j}^{(1)} \cdot \prod_{k=2}^{\ell} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}}. \tag{2}$$

Example *Consider the path* $\pi_1 = [(\mathsf{A}^{\mathbf{1}}, \mathsf{a}, \mathsf{b}), (\mathsf{A}, \mathsf{c}, \mathsf{d}), (\mathsf{B}, \mathsf{e}, \mathsf{f}), (\mathsf{C}, \mathsf{g}, \mathsf{h})]$ *in Fig. 1a. Then, Eq.* (2) *results in an allocation* $\mathcal{A}_1(\pi_1) = 1 \cdot \frac{1}{2} \cdot \frac{2}{4} \cdot \frac{1}{4} = \frac{1}{16}$.

To understand the idea behind this formula we consider some node k with interface i, connected through this interface to a neighboring node n. If node n can guarantee that the sum the preliminary allocations of all preliminary paths going towards node k is at most $DIV_i^{(k)}$, then $\mathcal{A}_1$ ensures that for each of node k's interfaces j, the sum of all preliminary allocations of all preliminary paths going through (i, j) is at most $M_{i,j}^{(k)}$. If all neighbors can provide this guarantee, no pair allocation of node k will be over-allocated, which implies that also none of its convergents will be over-allocated. If node k's convergents are smaller or equal to the corresponding divergents of its neighbors, also node k can give this guarantee to all of its neighbors. Therefore $\mathcal{A}_1$ will never cause over-allocation, if *every node's convergents are smaller or equal to the corresponding divergents of its neighbors*—which is an assumption we want to get rid of.

Example *The graph in Fig. 1a ensures that the divergent of a node is always larger than the convergent of the previous node when going* upwards. *Going* downwards, *this is not the case. Indeed, already two paths* $\pi_2 = [(\mathsf{B}, \mathsf{r}, \mathsf{e}), (\mathsf{A}, \mathsf{d}, c), (\mathsf{A}^{\mathbf{1}}, \mathsf{b}, \mathsf{a})]$ *with* $\mathcal{A}_1(\pi_2) = 2 \cdot \frac{1}{2} \cdot \frac{1}{1} = 1$ *and* $\pi_3 = [(\mathsf{C}, \mathsf{h}, \mathsf{g}), (\mathsf{B}, \mathsf{f}, \mathsf{e}), (\mathsf{A}, \mathsf{d}, c), (\mathsf{A}^{\mathbf{1}}, \mathsf{b}, \mathsf{a})]$ *(reverse of* π_1*) with* $\mathcal{A}_1(\pi_3) = 1 \cdot \frac{2}{4} \cdot \frac{1}{2} \cdot \frac{1}{1} = \frac{1}{4}$ *together cause an over-allocation of interface pairs* (d, c) *and* (b, a).

3.2 Step 2: a general solution for no-over-allocation

As over-allocation with $\mathcal{A}_1$ can only occur when some node's convergent is larger than the corresponding divergent of its neighbor, we can normalize each preliminary allocation to compensate this disparity. More concretely, if $CON_i^{(k-1)} > DIV_j^{(k)}$ for an on-path node k, the preliminary allocation from node $k-1$ is multiplied with:

$$\frac{DIV_j^{(k)}}{CON_i^{(k-1)}} \cdot \frac{M_{i,j}^{(k)}}{DIV_j^{(k)}} = \frac{M_{i,j}^{(k)}}{CON_i^{(k-1)}}. \tag{3}$$

Adapting Eq. (2) to this modification gives rise to the following formula:

$$\mathcal{A}_2(\pi) = M_{i,j}^{(1)} \cdot \prod_{k=2}^{\ell} \frac{M_{i,j}^{(k)}}{\max\{CON_j^{(k-1)}, DIV_i^{(k)}\}}. \tag{4}$$

Example *We find* $\mathcal{A}_2(\pi_3) = 1 \cdot \frac{2}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{16} = \mathcal{A}_2(\pi_1)$*;* $\mathcal{A}_2(\pi_2) = 2 \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{4}$.

This algorithm will never cause over-allocation, which follows directly from our proof in §4.1. Unfortunately, $\mathcal{A}_2$ is neither monotonic nor Pareto optimal. We can see why this is the case by taking a closer look at the contribution of some node k to the calculated allocations, which consists of the values ($DIV_i^{(k)}, M_{i,j}^{(k)}$, $CON_j^{(k)}$). In Eq. (4), the only subterm depending on those values is

$$\frac{M_{i,j}^{(k)}}{\max\{CON_j^{(k-1)}, DIV_i^{(k)}\} \cdot \max\{CON_j^{(k)}, DIV_i^{(k+1)}\}}. \tag{5}$$

Increasing $M_{i,j}^{(k)}$ by $\delta > 0$, and thus, implicitly, also $DIV_i^{(k)}$ and $CON_j^{(k)}$ by δ, can potentially contribute twice to the denominator and only once to the nominator of Eq. (5), thereby reducing all the allocations going through the interface (i, j).

Example *Consider increasing the pair allocation* (c, d) *to* $\widetilde{M}_{\mathsf{c},\mathsf{d}}^{(\mathsf{A})} = 9$*, leaving everything else unchanged. Then,* $\widetilde{\mathcal{A}}_2(\pi_2) = 2 \cdot \frac{9}{10} \cdot \frac{1}{10} = \frac{18}{100} < \frac{1}{4} = \mathcal{A}_2(\pi_2)$.

In general, $\mathcal{A}_2$ provides suboptimal allocations when there is a node k with "*superfluous* allocations", i.e., where $DIV_i^{(k)} > CON_j^{(k-1)}$ and $CON_j^{(k)} > DIV_i^{(k+1)}$. We explain how to strictly improve this and present GMA in the next section.

3.3 Step 3: monotonic and Pareto-optimal allocations

The main idea to resolve the violation of monotonicity and optimality is to implicitly *scale down* the three-tuple of a node k with superfluous allocations to $(s \cdot DIV_i^{(k)}, s \cdot M_{i,j}^{(k)}, s \cdot CON_j^{(k)})$ for $0 < s < 1$, such that either $s \cdot DIV_i^{(k)} \leq CON_j^{(k-1)}$ or $s \cdot CON_j^{(k)} \leq DIV_i^{(k+1)}$. The intuition is that a third algorithm, based on $\mathcal{A}_2$ but with scaled-down three-tuples, does not cause over-allocation while observing monotonicity. We will prove later in §4 that this statement holds.

For some arbitrary path, we now want to find a way to optimally scale down the three-tuple $(DIV_i^{(k)}, M_{i,j}^{(k)}, CON_j^{(k)})$ of each node k. The result is a new algorithm that takes the original inputs, scales them down implicitly, and finally uses $\mathcal{A}_2$ to compute the allocation.

As we prove in Appendix B, down-scaling improves the resulting path allocation only for the case—as considered above—in which superfluous allocations are present ($DIV_i^{(k)} > CON_j^{(k-1)}$ and $CON_j^{(k)} > DIV_i^{(k+1)}$).[7] It is therefore sufficient to scale down the divergent of node k to the convergent of node $k-1$,

[7] $CON_j^{(k-1)}$ and $DIV_i^{(k+1)}$ might have already been scaled down.

any further scaling will not improve the allocation. This observation results in the following iterative algorithm.

On a path π with ℓ nodes, we start from node 1. As there is no previous node, scaling is not possible, and the scaling factor is $f^{(1)} = 1$. At the second node, the convergent of the first node can either be smaller than the divergent of the second node, or larger. In the first case, we scale down the three-tuple of the second node by $CON_j^{(1)}/DIV_i^{(2)}$. In the second, no scaling down is possible. In both cases we thus scale down the three-tuples of node 2 by $f^{(2)} = \min\{1, CON_j^{(1)}/DIV_i^{(2)}\}$, and so the first factor of the product in Eq. (4) becomes

$$\frac{M_{i,j}^{(2)} \cdot f^{(2)}}{\max\{CON_j^{(1)}, DIV_i^{(2)} \cdot f^{(2)}\}} = \frac{M_{i,j}^{(2)} \cdot f^{(2)}}{CON_j^{(1)}}. \tag{6}$$

At the third node this case distinction is repeated. However, recall that the convergent of the second node might have been scaled down, so we have to use the value $(f^{(2)} \cdot CON_j^{(2)})$ instead of $CON_j^{(2)}$ in the computation. Therefore, taking $f^{(3)} = \min\{1,\ (CON_j^{(2)} \cdot f^{(2)})/DIV_i^{(3)}\}$, we obtain the third factor of the product in Eq. (4):

$$\frac{M_{i,j}^{(3)} \cdot f^{(3)}}{\max\{CON_j^{(2)} \cdot f^{(2)}, DIV_i^{(3)} \cdot f^{(3)}\}} = \frac{M_{i,j}^{(3)} \cdot f^{(3)}}{CON_j^{(2)} \cdot f^{(2)}}. \tag{7}$$

Continuing this expansion, we can define the scaling factors f recursively for each node as

$$f^{(1)} = 1; \qquad f^{(k)} = \min\left\{1,\ \frac{CON_j^{(k-1)} \cdot f^{(k-1)}}{DIV_i^{(k)}}\right\}. \tag{8}$$

Overall, we modify Eq. (4) in the following way:

$$\mathcal{G}(\pi) = M_{i,j}^{(1)} \cdot \prod_{k=2}^{\ell} \frac{M_{i,j}^{(k)} \cdot f^{(k)}}{CON_j^{(k-1)} \cdot f^{(k-1)}} = f^{(\ell)} \cdot \frac{\prod_{k=1}^{\ell} M_{i,j}^{(k)}}{\prod_{k=2}^{\ell} CON_j^{(k-1)}}, \tag{9}$$

which is equivalent to computing $\mathcal{A}_2$ on the scaled-down input three-tuples. The last step follows from rearranging indices and realizing that $f^{(k)}$ can be factored out recursively, apart from the first ($f^{(1)} = 1$) and the last one. Instead of this recursive formulation, Eq. (9) can also be written as a direct formula (the proof can be found in Appendix C).

The *global myopic allocation (GMA) algorithm*:

$$\mathcal{G}(\pi) = \min_x \left(\prod_{k=1}^{x-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(x)} \cdot \prod_{k=x+1}^{\ell} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \right) \tag{10}$$

Example *Consider again our example of Fig. 1a with* $\widetilde{M}_{\mathsf{c,d}}^{(\mathsf{A})} = 9$. *In this case we have* $DIV_{\mathsf{d}}^{(\mathsf{A})} = 10 > CON_{\mathsf{e}}^{(\mathsf{B})} = 4$ *and* $CON_{\mathsf{c}}^{(\mathsf{A})} = 10 > DIV_{\mathsf{b}}^{(\mathsf{A}^1)} = 1$. *The three-tuple of* A *can thus be scaled down by a factor of* $\frac{4}{10}$. *Using Eq.* (10)

for the path π_2, we find that the argument of the minimum is $\mathsf{A}^{\mathbf{1}}$ *and* $\mathcal{G}(\pi_2) = \frac{2}{4} \cdot \frac{9}{10} \cdot 1 = \frac{9}{20} > \frac{18}{100} = \widetilde{\mathcal{A}}_2(\pi_2)$.

4 Proofs of GMA's properties

In this section, we prove that GMA's computation described in Eq. (10) satisfies the properties defined in §2. We prove the core property C1 in §4.1 and Opt in §4.2. Locality (C2) follows directly from Eq. (10), as the computation only involves allocation-matrix entries of the nodes on the path. The supplementary properties S1–S3 are proven in Appendix E.

4.1 Proof of no-over-allocation (C1)

In this subsection we prove that there is no resource overuse of any of the pair allocation $M_{i,j}^{(k)}$, which, by the fact that convergent and divergent of an interface must be smaller than the capacity of the edge connected to it, implies that there is also no overuse on any edge of the graph. In the context of this proof, the $+$ operator is not only used for addition, but also for list concatenation. We denote the set of non-local interfaces of some node k as $I_{\text{ext}}^{(k)}$. We will use the notation $M_{i,j}^{(k)}(\pi)$ to state more precisely which path the variable refers to. We want to prove that for every node k and all of its interface pairs, the corresponding pair allocation is greater than or equal to the sum of all resource allocations of all paths going through that interface pair. For this we distinguish the following cases an interface pair can be assigned to, and prove each case individually:
Case 1: The interface pair starts from a local interface: $(\perp, j)$
Case 2: The interface pair ends in a local interface: $(i, \perp)$
Case 3: The interface pair starts and ends in non-local interfaces: (i, j)

Case 1: We will prove a stronger statement, captured by the following lemma:

Lemma 1. *For an arbitrary node A and an arbitrary non-local interface j^A, let S_{t}^x be the set of* terminated *paths of length* at most *x that start in $(\perp, j^A)$, and S_{p}^x the set of* preliminary *paths of length* exactly *x that start in $(\perp, j^A)$. Then*

$$\forall x \geq 1 : \sum_{\pi \in S_{\text{p}}^x} \mathcal{G}(\pi) + \sum_{\pi \in S_{\text{t}}^x} \mathcal{G}(\pi) \leq M_{\perp,j}^{(A)}. \tag{11}$$

We emphasize that, by the definition in Eq. (10), GMA not only allows to calculate allocations on terminated, but also on preliminary paths. The lemma implies our original statement, i.e., $\forall x \geq 1$: $\sum_{\pi \in S_{\text{t}}^x} \mathcal{G}(\pi) \leq M_{\perp,j}^{(A)}$.

Proof. We prove Lemma 1 by induction over x for arbitrary A and j^A.

Base case ($x = 1$): We have $S_{\text{p}}^1 = \{\ [(\perp, j^A)]\ \}$ and $S_{\text{t}}^1 = \{\}$, which directly implies $\sum_{\pi \in S_{\text{p}}^1} \mathcal{G}(\pi) + \sum_{\pi \in S_{\text{t}}^1} \mathcal{G}(\pi) = M_{\perp,j}^{(A)} \leq M_{\perp,j}^{(A)}$.

Inductive step:

Induction hypothesis: For a particular x: $\sum_{\pi\in S_\mathrm{p}^x}\mathcal{G}(\pi)+\sum_{\pi\in S_\mathrm{t}^x}\mathcal{G}(\pi)\leq M_{\perp,j}^{(A)}$.

To show: $\sum_{\pi\in S_\mathrm{p}^{x+1}}\mathcal{G}(\pi)+\sum_{\pi\in S_\mathrm{t}^{x+1}}\mathcal{G}(\pi)\leq M_{\perp,j}^{(A)}$.

Definitions: For some preliminary path π of length ℓ, let node Z be the node that is connected to j^ℓ and let the corresponding interface of Z be i^Z. We define the *local extension of a path* π as $E_\mathrm{loc}(\pi):=\{\ \pi+[(i^Z,\perp)]\ \}$, the *non-local extension of a path* π as $E_\mathrm{ext}(\pi):=\cup_{j^z\in I_\mathrm{ext}^{(Z)}}\{\ \pi+[(i^Z,j^Z)]\ \}$ and their union as $E(\pi):=E_\mathrm{loc}(\pi)\cup E_\mathrm{ext}(\pi)$.

Proof:

$$\sum_{\pi\in S_\mathrm{p}^{x+1}}\mathcal{G}(\pi)+\sum_{\pi\in S_\mathrm{t}^{x+1}}\mathcal{G}(\pi)\ =\left(\sum_{\pi\in S_\mathrm{p}^x}\sum_{\phi\in E_\mathrm{ext}(\pi)}\mathcal{G}(\phi)\right)+\left(\sum_{\pi\in S_\mathrm{t}^x}\mathcal{G}(\pi)+\sum_{\pi\in S_\mathrm{p}^x}\sum_{\phi\in E_\mathrm{loc}(\pi)}\mathcal{G}(\phi)\right) \tag{12a}$$

$$=\sum_{\pi\in S_\mathrm{p}^x}\sum_{\phi\in E(\pi)}\mathcal{G}(\phi)+\sum_{\pi\in S_\mathrm{t}^x}\mathcal{G}(\pi) \tag{12b}$$

$$=\sum_{\pi\in S_\mathrm{p}^x}\sum_{\phi\in E(\pi)}\min\left(\mathcal{G}(\pi)\cdot\frac{M_{i,j}^{(Z)}}{DIV_i^{(Z)}},\ \prod_{k=1}^{\ell}\frac{1}{CON_j^{(k)}}\cdot M_{i,j}^{(Z)}\right)+\sum_{\pi\in S_\mathrm{t}^x}\mathcal{G}(\pi) \tag{12c}$$

$$\leq\sum_{\pi\in S_\mathrm{p}^x}\sum_{\phi\in E(\pi)}\frac{M_{i,j}^{(Z)}}{DIV_i^{(Z)}}\cdot\mathcal{G}(\pi)+\sum_{\pi\in S_\mathrm{t}^x}\mathcal{G}(\pi)=\sum_{\pi\in S_\mathrm{p}^x}\mathcal{G}(\pi)+\sum_{\pi\in S_\mathrm{t}^x}\mathcal{G}(\pi)\ \leq\ M_{\perp,j}^{(A)} \tag{12d}$$

In the step from Eq. (12b) to Eq. (12c), we used the fact that when extending the path, the argument of the minimum of Eq. (10) either stays the same, or the newly added node now minimizes the formula, which follows directly from Eq. (9). The transition in Eq. (12d) follows from $\sum_{\phi\in E(\pi)}M_{i,j}^{(Z)}=DIV_i^{(Z)}$.

Case 2: The proof is exactly the same as for case 1, except that we extend the path in the backward instead of the forward direction. The only change required is the adaptation of the definitions of local and non-local extensions of a path and we use $\sum_{\phi\in E(\pi)}M_{i,j}^{(Z)}=CON_j^{(Z)}$.

Case 3: Choose an arbitrary node A. Then choose arbitrary non-local interfaces $i^A,j^A\in I_\mathrm{ext}^{(A)}$ of node A. Using exactly the same procedure as for the proof of case 2, but using (i^A,j^A) as the interface pair where the paths "end" (it does not terminate in a local interface), we can show that the sum of all resource allocations for all paths *ending* in (i^A,j^A) is always smaller or equal to $M_{i,j}^{(A)}$. We then choose an arbitrary path π that ends in (i^A,j^A). Using the same procedure as for the proof of case 1, but using (i^A,j^A) as the interface pair where the paths "begin" (it does not start in a local interface) and setting $\hat{M}_{i,j}^{(A)}:=\mathcal{G}(\pi)$, we can show that the sum of the resource allocations of all the (terminated) paths that

extend π never exceeds $\mathcal{G}(\pi)$. It follows that the sum of the resource allocations of all the paths going through (i^A, j^A) never exceeds $M_{i,j}^{(A)}$. □

4.2 Proof of optimality (Opt)

In this section we show that GMA is optimal according to Opt, which means that there is no better local (C2) algorithm that does not over-allocate any edge or interface pair (C1). As every invocation of a local algorithm is only based on the nodes of one path, and is oblivious of all the other nodes of the graph, in order to prevent overuse the algorithm has to consider all possible graphs containing this path. This insight is central for the proof of optimality and is formalized in the following lemma:

Lemma 2. *For every allocation graph and every one of its paths π, there exists another allocation graph that contains a path with the same sequence of allocation matrices, where the pair allocation $M_{i,j}^{(x)}$ of some on-path node x is fully utilized (there is no available resource left) if there is a GMA allocation on every path containing (x, i^x, j^x) in this new graph.*

Proof. Let π be an arbitrary path of an arbitrary allocation graph, and let x be the index for which Eq. (10) is minimized. We construct a new allocation graph around π as follows:

- Remove all the nodes that are not part of π.
- Keep the on-path nodes, their interfaces, and their allocation matrices as they are.
- For every node, create identical copies of the node for each of its occurrences on the path (multiple copies, in case the path contains loops) and only keep the edges to the previous and subsequent node on the path.
- For all these on-path nodes, attach new nodes to the non-local interfaces that are not already part of π. Those new nodes only have one local and one non-local interface (the interface through which they are attached to the on-path node).
- For every node $k \in \{1, \ldots, x-1\}$ and each of its interfaces $\widetilde{i}$ to which a new node was attached, the pair allocation (from its local to its non-local interface) of the new node is set to $DIV_{\widetilde{i}}^{(k)}$. This implies that also the divergent (at the local interface) and the convergent (at the non-local interface) of the new node are equal to $DIV_{\widetilde{i}}^{(k)}$.
- For x, the newly attached nodes can have arbitrary allocation-matrix entries.
- For every node $k \in \{x+1, \ldots, \ell\}$ and each of its interfaces $\widetilde{j}$ to which a new node was attached, the pair allocation (from its non-local to its local interface) of the new node is set to $CON_{\widetilde{j}}^{(k)}$. This implies that also the divergent (at the non-local interface) and the convergent (at the local interface) of the new node are equal to $CON_{\widetilde{j}}^{(k)}$.

Given that there is a GMA allocation on every possible path (in our new graph) going through (i^x, j^x), we want to show that $M_{i,j}^{(x)}$ is fully utilized. We characterize all possible paths for three cases: If $1 < x < \ell$ (case 1), a path starts at a local interface of some node $k \leq x-1$ or at the local interface of some of its attached nodes, and ends at a local interface of some node $m \geq x+1$ or at the local interface of some of its attached nodes. If $x = 1$ (case 2), every path starts at the local interface of x, and ends at a local interface of some node $k \geq 2$ or at the local interface of some of its attached nodes. If $x = \ell$ (case 3), every path starts at a local interface of some node $k \leq \ell - 1$ or at the local interface of some of its attached nodes, and ends at the local interface of node ℓ.

Case 1: We use the following notation in order to simplify our proof:

$$a^{(u)} = \frac{M_{i,j}^{(u)}}{CON_j^{(u)}}, \; b^{(u)} = \frac{M_{i,j}^{(u)}}{DIV_i^{(u)}} \tag{13}$$

Let R_u be the sum of all allocations of all the nodes $k \in \{1, \ldots, x-1\}$ starting either at a local interface or at the local interface of some of its attached nodes, and ending either at a local interface of node u or at the local interface of some of its attached nodes, divided by $M_{i,j}^{(x)}$. Thus, we need to prove

$$M_{i,j}^{(x)} \cdot \sum_{u=x+1}^{\ell} R_u = M_{i,j}^{(x)} \quad \Leftrightarrow \quad \sum_{u=x+1}^{\ell} R_u = 1. \tag{14}$$

We formulate two lemmas, which are proven in Appendix D:

Lemma 3. *For $a_1, \ldots, a_x > 0$: $\prod_{i=1}^{x} a_i + \sum_{k=1}^{x} \left((1-a_k) \cdot \prod_{i=k+1}^{x} a_i \right) = 1$.*

Lemma 4. *$R_\ell = \prod_{k=x+1}^{\ell-1} b^{(k)}$ and $R_u = (\prod_{k=x+1}^{u-1} b^{(k)}) \cdot (1 - b^{(u)})$ (for $x+1 \leq u \leq \ell - 1$).*

These lemmas immediately imply our proof goal:

$$\sum_{u=x+1}^{\ell} R_u = \sum_{u=x+1}^{\ell-1} R_u + R_\ell = \sum_{u=x+1}^{\ell-1} \prod_{k=x+1}^{u-1} b^{(k)} \cdot (1 - b^{(u)}) + \prod_{k=x+1}^{\ell-1} b^{(k)} = 1. \tag{15}$$

Case 2+3: The proofs follow a simplified structure of the proof of case 1. □

Theorem 5. *GMA is Pareto optimal among all algorithms in the sense of Opt.*

Proof. This follows directly from Lemma 2: for a given path (nodes with their associated allocation matrices) there always exists a graph containing that path, where increasing the allocation calculated by GMA will cause overuse, which can only be prevented by decreasing allocations on other paths. □

5 GMA provides meaningful allocations

A potential limitation of GMA is the size of the allocations it provides. We proved that GMA's path allocations are small enough that, even if all the paths

have an allocation, no over-allocation occurs. In this section we show that GMA's path allocations are still large enough to satisfy the requirements of the critical applications that motivate this work (details in Appendix A). We do this by simulating GMA on random graphs, thereby exploring the trade-offs between graph topology and the resulting GMA allocation sizes.

5.1 Simulation setup

Graph topology. We use the well-known Barabási–Albert random graph model to generate allocation graphs [2]. This algorithm is designed to produce scale-free random graphs, which are found to well approximate real-life technological networks [6].

At the topological level, the size of a GMA allocation for some path depends on (i) the degree of the nodes on the path, as it determines the size of the allocation matrix, (ii) the length of the path, since Eq. (10) contains an iterative product on each node on the path, and (iii) the capacity of each on-path edge (discussed in the next paragraph). We aggregate the first two metrics at the graph level by considering the average node degree and the diameter of the graph, i.e., the length of the longest path.[8] Therefore, we generate 275 random graphs for our simulations, with 8 to 2048 nodes, varying average degree and diameter. Additional details on graph generation can be found in Appendix G.

Resources and Allocation matrices. In the simulations, we model the varying bandwidth of real-world network links by assigning different capacities to the edges of graphs. To assign capacity to edges based on a *degree–gravity* model: the capacity of a (directed) edge is selected proportionally to the product of the degrees of its adjacent nodes [14]. We discretize these values to 10 different levels from 40 to 400. This choice is motivated by real networks, where more connected nodes also tend to have higher forwarding capabilities.

Based on these edge capacities, we then create the allocation matrices. Although each node might have different policies, simulating those policies for the nodes introduces many additional degrees of complexity, beyond the scope of this evaluation. Therefore, we assume a simple *proportional sharing* policy to construct an allocation matrix, which we obtain by performing the following three steps for each node k and all its interfaces i and j: (i) $M_{i,j}^{(k)} \leftarrow cap_i^{(k)}$, while for the local interface $\perp$, $M_{\perp,j}^{(k)}, M_{i,\perp}^{(k)} \leftarrow \max_i\{cap_i^{(k)}\}$; (ii) $M_{i,j}^{(k)} \leftarrow M_{i,j}^{(k)} \cdot cap_j^{(k)}/CON_j^{(k)}$; (iii) if $DIV_i^{(k)} > cap_j^{(k)}$, then $M_{i,j}^{(k)} \leftarrow M_{i,j}^{(k)} \cdot cap_i^{(k)}/DIV_i^{(k)}$.

Path selection. In this simulation, the goal is to create path allocations between every pair of nodes. Motivated again by networking practice, we consider allocations made on k-shortest paths, with $k \in \{1, 2, 3\}$. For $k = 1$, we create allocations on the single-shortest path for every pair of nodes. However, GMA

[8] These two factors are closely related with each other and to the number of nodes in the graph: keeping the number of nodes fixed, a graph with higher average node degree will inevitably have smaller diameter.

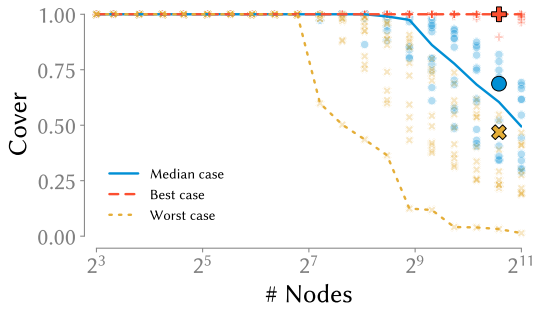


Figure 2: ***Minimum, maximum, and median single-path*** $\mathbf{10^{-4}}$***-cover.*** *The highlighted markers show the max ✚, median ●, and min ✖ cover for one specific graph (which is further analyzed in Figs. 5 and 6 in Appendix G).*

can compute an allocation for *any* path in the graph. Therefore, if two nodes are able to use multiple paths simultaneously, the total allocation for the pair is the aggregate of the allocations on the individual paths. We then create allocations on the 2- and 3-shortest paths for every pair of nodes, and evaluate the advantage that multipath communication can provide.

Metrics: α-cover. Given a source node, the size of the GMA allocations to different destination nodes can vary greatly, and computing average statistics does not reflect the binary nature of critical application requirements: either the allocation exceeds the minimum usability threshold, or the allocation is not useful (see Appendix A for details).

Therefore, we introduce a new metric to aggregate this information and compare the effectiveness of GMA across different topologies, called α-cover. Given a source node in a graph and a path selection strategy, the node's α-cover is the fraction of destination nodes to which the sum of the path allocations computed over the available paths is more than α. Therefore, α-cover captures the size of the sub-graph with which the source node can communicate using an adequately-sized GMA allocation. For example, a node with a 10^{-4}-cover of 0.7 can reach 70 % of the nodes in the graphs with an allocation of at least 10^{-4}. Naturally, higher values of α-cover are better. We define the median α-cover of a graph as the median of the α-covers of its nodes (and similarly for minimum and maximum). While different applications will require different values of α, we use a 10^{-4}-cover in all simulations. Again, this is motivated by practical considerations: if we set 1 unit of resource = 1 Gbps, 10^{-4} units correspond to 100 kbps. The applications that motivate this work, such as blockchains and inter-bank transaction clearing, can comfortably operate within this boundary.

5.2 Results

For each of the generated graphs, Fig. 2 relates its minimum, maximum, and median 10^{-4}-cover to the number of nodes, where we used the single shortest path selection scheme. We see that all graphs have a median cover in the upper

50 % range , while the minimum cover decreases to just a few percent for graphs with a high number of nodes. Graphs with lower median cover are the ones that have low or high diameter, as Fig. 5 in Appendix G shows. This confirms the observation that large allocation matrices (low diameter) or long paths (high diameter) decrease the size of allocations. Further, in all graphs, we find at least one node with cover greater than 89 %, and observe that the cover increases with the degree of the nodes: central nodes have therefore better cover, an important property in practical applications. An example is shown in Fig. 5 in Appendix G.

Figure 3 in Appendix G shows the improvement in the median cover of the graphs when using the 2- or 3-shortest path selection schemes in place of of the single shortest path selection scheme. We see that the returns for using additional paths are high, reaching over 120 % increase over single-path cover when using three paths instead of one. Graphs with lower number of nodes benefit less from the additional paths, as many already achieve perfect cover. A higher k could further increase the cover, but this exploration is left to future work.

6 Related work

Flow problems and algorithms. A class of theoretical problems that are related to our path-allocation problem are *multi-commodity flow problems*, which have been studied extensively since the 1950s [9]. The variant which is most closely related to our setting is the *maximum concurrent flow problem* [16], where fairness between different commodities is taken into account, but the ratios are set by a central controller. All variants differ from our PA^{dist} problem in that they (i) do not consider independent nodes with their own properties and (ii) require a global knowledge of the topology. They have thus been applied mostly to centrally controlled networks [8].

Resource allocation in networks. Bandwidth guarantees were a central concept of virtual-circuit architectures like ATM [15]. For today's IP-based Internet, bandwidth reservations have been proposed in the Integrated Services (IntServ) architecture [4], in which they are negotiated through the Resource Reservation Protocol (RSVP) [5]. However, due to its high reliance on in-network state, IntServ has never been widely adopted. Further, these systems do not specify *how much* bandwidth should be allocated to flows. The Internet overwhelmingly relies on congestion control [10, 12] as a distributed mechanism for bandwidth allocation between flows, which provides no guarantees to the communication partners and has no support to implement traffic policies. There exists a wide range of traffic-engineering systems suitable to intra-domain contexts, such as MPLS [13] with OSPF-TE [11] and RSVP-TE [1] or SDN-based solutions [17]. However, in contrast to GMA, which supports autonomous nodes, all these systems require a central controller.

7 Discussion and Conclusion

In this paper, we revisit an old networking and distributed-systems problem—how to allocate resources in a network of independent nodes when no central controller is available. After introducing the formalism of allocation graphs, in which each node is associated with *local* allocations based on available resources and policies, we ask a novel question: can an algorithm compute resource allocations for all paths in an allocation graph, without causing over-allocation, and relying only on local information? This is the foundation of the PA^{dist} problem. We answer with our *global myopic allocation* (GMA) algorithm, showing how these local decisions give rise to meaningful and sustainable *global* allocations. Further, we prove that these allocations are Pareto-optimal, and therefore cannot be trivially improved.

Relevance to networking. The allocations calculated through GMA are static and depend only on the policies of on-path nodes; in particular, they are independent of other allocations and resource demands. They thus provide strong minimal resource guarantees that are valid under all networking conditions and are particularly relevant for applications where centralized solutions based on dedicated network infrastructure are too expensive or inherently impossible. By their very nature, these guaranteed allocations are smaller than what can be achieved through dynamic resource-allocation systems. However, our simulations show that, even under conservative assumptions, GMA provides sufficient communication bandwidth to virtually all pairs of nodes in small to medium-sized networks. Thus, GMA-based allocations with strong availability guarantees could *complement* other systems with higher network utilization but weaker guarantees, such as best-effort traffic.

Future work. The novel results on graph resource allocation presented in this paper open many new and exciting avenues for future research, both theoretical and applied. First of all, this paper did not explore the *fairness* implications of GMA allocations. The properties of monotonicity and Pareto-optimality, along with the proportional use of pair allocations in the computation, point towards a strong *neighbor-based* fairness notion. We leave the analysis of such a notion to future work. Second, we see great potential for further research on PA^{dist} algorithms. For instance, Pareto optimality does not satisfy the question of whether GMA is optimal in a global sense, i.e., whether it maximizes a function over all path allocations—their sum, for example. The discovery of globally optimal PA^{dist} algorithms could lead to interesting theoretical advancements, with profound practical implications.

Finally, in this paper we have discussed how allocations can be *computed* in a distributed setting. This is orthogonal to the development of specific protocols necessary to communicate and authenticate necessary information and enforce the allocations. Future research could focus on the development of such a protocol and investigate its interplay with other networking paradigms like best-effort traffic and congestion control.

Acknowledgments

We would like to thank Mohsen Ghaffari for the illuminating discussions; Tobias Klenze, Simon Scherrer, Stefan Schmid, and Joel Wanner for their feedback on the manuscript; and the anonymous reviewers for their insightful comments.

References

1. Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., Swallow, G.: RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209, IETF (2001)
2. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439) (1999)
3. Basescu, C., Reischuk, R.M., Szalachowski, P., Perrig, A., Zhang, Y., Hsiao, H.C., Kubota, A., Urakawa, J.: SIBRA: Scalable Internet bandwidth reservation architecture. In: NDSS (2016)
4. Braden, R., Clark, D., Shenker, S.: Integrated Services in the Internet Architecture: an Overview. RFC 1633, IETF (1994)
5. Braden, R., Zhang, L., Berson, S., Herzog, S., Jamin, S.: Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. RFC 2205, IETF (1997)
6. Broido, A.D., Clauset, A.: Scale-free networks are rare. Nature Communications **10**(1) (2019)
7. Brown, L., Ananthanarayanan, G., Katz-Bassett, E., Krishnamurthy, A., Ratnasamy, S., Schapira, M., Shenker, S.: On the future of congestion control for the public internet. In: ACM HotNets (2020)
8. Chang, T., Tang, Y., Chen, Y., Hsu, W., Tsai, M.: Maximum concurrent flow problem in MPLS-based software defined networks. In: IEEE Global Communications Conference (GLOBECOM) (2018)
9. Ford Jr, L.R., Fulkerson, D.R.: A suggested computation for maximal multi-commodity network flows. Management Science **5**(1) (1958)
10. Jacobson, V.: Congestion avoidance and control. SIGCOMM CCR **18**(4) (1988)
11. Katz, D., Kompella, K., Yeung, D.: Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630, IETF (2003)
12. Kelly, F.P., Maulloo, A.K., Tan, D.K.: Rate control for communication networks: shadow prices, proportional fairness and stability. Journal of the Operational Research society **49**(3) (1998)
13. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031, IETF (2001)
14. Saino, L., Cocora, C., Pavlou, G.: A toolchain for simplifying network simulation setup. In: International Conference on Simulation Tools and Techniques (2013)
15. Saitō, H.: Teletraffic Technologies in ATM Networks. Artech House (1994)
16. Shahrokhi, F., Matula, D.W.: The maximum concurrent flow problem. Journal of the ACM **37**(2) (1990)
17. Shu, Z., Wan, J., Lin, J., Wang, S., Li, D., Rho, S., Yang, C.: Traffic engineering in software-defined networking: Measurement and management. IEEE Access **4** (2016)
18. Ware, R., Mukerjee, M.K., Seshan, S., Sherry, J.: Beyond Jain's fairness index: Setting the bar for the deployment of congestion control algorithms. In: ACM HotNets (2019)

A Critical networking applications

For many *critical* applications, reliability, security, and scalability of communication systems are of paramount importance. These application require relatively low traffic volumes, but availability has to be guaranteed at all times for these services to achieve their task. We provide two examples of such applications.

The first is inter-bank transactions. The SWIFT financial messaging network is a prominent example in this market, as it handles transactions between its 11 000 member institutions and accounts for half of global cross-border inter-bank transactions [2]. Despite the importance of these transactions for today's financial system, their actual bandwidth requirements are modest. On an average day, SWIFT processes around 40 million messages in total [6], which corresponds to fewer than 500 messages per second—globally. Each transaction is encoded in an XML file of variable size, usually around a few kilobytes (estimate based on real-world examples of the XML-encoded ISO 20022 transaction message format [5]), resulting on an average load of less than 1 Mbps between all 11 000 institutions.

The Bitcoin network provides a second example. Each Bitcoin miner node needs to run the consensus protocol in order to verify the transaction that are being committed to the blockchain. Today, the network processes 7 transactions per second [3], with an average transaction size of 500 B, and very rarely above 1 kB [7, 8]. This directly translates to modest bandwidth requirements of less than 100 kbps per node. However, delays or interruptions of communication can result in financial loss [1]. A further complication complication that arises in blockchain networks is decentralization. As nodes are run by different—and often untrusted—entities, centralized solutions are avoided as they introduce a single point of failure. Even permissioned blockchains, like the Libra network, impose node decentralization by design as a way to build trust [4].

In general, critical applications share these common traits: (i) the required traffic volumes are relatively small, less than 100 kbps per end-to-end communication, but (ii) connectivity has to be ensured at all times (availability), (iii) even in the presence of denial-of-service (DoS) attacks (security). Finally, (iv) the guarantees have to be extended to large networks, in many cases under the assumption of decentralized control.

B Cases in which down-scaling improves the allocation calculated by Eq. (4)

Lemma 6. *Let π be an arbitrary path consisting of ℓ nodes, and let k be one if its on-path nodes. If $1 < k < \ell$, scaling down its contributed values $(DIV_i^{(k)}, M_{i,j}^{(k)}, CON_j^{(k)})$ (without scaling down any values of other nodes) can only improve $\mathcal{A}_2(\pi)$ if $DIV_i^{(k)} > CON_j^{(k-1)}$ and $CON_j^{(k)} > DIV_i^{(k+1)}$.*
If $k = 1$ or $k = \ell$, scaling-down its hop values will never increase $\mathcal{A}_2(\pi)$.

Proof.
Case $1 < k < \ell$**:** The contributed values of node k are part of the following factor of Eq. (4):

$$F := \frac{M_{i,j}^{(k)}}{\max\{CON_j^{(k-1)}, DIV_i^{(k)}\} \cdot \max\{CON_j^{(k)}, DIV_i^{(k+1)}\}}. \tag{16}$$

We write $\tilde{F}$ for the same factor after scaling down the values ($DIV_i^{(k)}, M_{i,j}^{(k)}$, $CON_j^{(k)}$).

Case $(DIV_i^{(k)} > CON_j^{(k-1)}) \wedge (CON_j^{(k)} > DIV_i^{(k+1)})$**:** Scaling down the contributed values by $s := CON_j^{(k-1)}/DIV_i^{(k)} < 1$ leads to

$$\tilde{F} = \frac{s \cdot M_{i,j}^{(k)}}{s \cdot DIV_i^{(k)} \cdot \max\{s \cdot CON_j^{(k)}, DIV_i^{(k+1)}\}} > \frac{M_{i,j}^{(k)}}{DIV_i^{(k)} \cdot CON_j^{(k)}} = F. \tag{17}$$

Case $(DIV_i^{(k)} > CON_j^{(k-1)}) \wedge (CON_j^{(k)} \leq DIV_i^{(k+1)})$**:** Scaling down the contributed values by s where $CON_j^{(k-1)}/DIV_i^{(k)} \leq s < 1$ has no impact on Eq. (16):

$$\tilde{F} = \frac{s \cdot M_{i,j}^{(k)}}{s \cdot DIV_i^{(k)} \cdot \max\{s \cdot CON_j^{(k)}, DIV_i^{(k+1)}\}} = \frac{M_{i,j}^{(k)}}{DIV_i^{(k)} \cdot DIV_i^{(k+1)}} = F. \tag{18}$$

Any further down-scaling only decreases the allocation, as shown in the last case.

Case $(DIV_i^{(k)} \leq CON_j^{(k-1)}) \wedge (CON_j^{(k)} > DIV_i^{(k+1)})$**:** The proof follows the same structure as in the previous case.

Case $(DIV_i^{(k)} \leq CON_j^{(k-1)}) \wedge (CON_j^{(k)} \leq DIV_i^{(k+1)})$**:** Scaling down the contributed values by any factor $s < 1$ leads to

$$\tilde{F} = \frac{s \cdot M_{i,j}^{(k)}}{\max\{CON_j^{(k-1)}, s \cdot DIV_i^{(k)}\} \cdot \max\{s \cdot CON_j^{(k)}, DIV_i^{(k+1)}\}} \tag{19a}$$

$$= \frac{s \cdot M_{i,j}^{(k)}}{CON_j^{(k-1)} \cdot DIV_i^{(k+1)}} < \frac{M_{i,j}^{(k)}}{CON_j^{(k-1)} \cdot DIV_i^{(k+1)}} = F. \tag{19b}$$

Case $k = \ell$**:** The contributed values of node ℓ are part of the following factor of Eq. (4):

$$\frac{M_{i,j}^{(\ell)}}{\max\{CON_j^{(\ell-1)}, DIV_i^{(\ell)}\}} \tag{20}$$

Scaling down the contributed values by any factor $s < 1$ modifies Eq. (20) to

$$\frac{s \cdot M_{i,j}^{(\ell)}}{\max\{CON_j^{(\ell-1)}, s \cdot DIV_i^{(\ell)}\}} = \frac{M_{i,j}^{(\ell)}}{\max\{\frac{1}{s} \cdot CON_j^{(\ell-1)}, DIV_i^{(\ell)}\}} \tag{21a}$$

$$\leq \frac{M_{i,j}^{(\ell)}}{\max\{CON_j^{(\ell-1)}, DIV_i^{(\ell)}\}}. \tag{21b}$$

Case $k = 1$: The proof follows the same structure as in the case $k = \ell$. □

C Equivalence of recursive and direct GMA formulas

Lemma 7. *Equation* (9) *is equivalent to Eq.* (10).

Proof. We prove Lemma 7 by induction over the path length ℓ.

Base case ($\ell = 1$): Because $f^{(1)} = 1$, we get $M_{i,j}^{(1)} = f^{(1)} \cdot M_{i,j}^{(1)}$.

Inductive step:

Induction hypothesis:

For a particular ℓ:

$$\frac{f^{(\ell)} \cdot \prod_{k=1}^{\ell} M_{i,j}^{(k)}}{\prod_{k=2}^{\ell} CON_j^{(k-1)}} = \min_{0 \leq x \leq \ell} \left(\prod_{k=1}^{x-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(x)} \cdot \prod_{k=x+1}^{\ell} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \right)$$

To show:

$$\frac{f^{(\ell+1)} \cdot \prod_{k=1}^{\ell+1} M_{i,j}^{(k)}}{\prod_{k=2}^{\ell+1} CON_j^{(k-1)}} = \min_{0 \leq x \leq \ell+1} \left(\prod_{k=1}^{x-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(x)} \cdot \prod_{k=x+1}^{\ell+1} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \right)$$

Proof:

$$\frac{f^{(\ell+1)} \cdot \prod_{k=1}^{\ell+1} M_{i,j}^{(k)}}{\prod_{k=2}^{\ell+1} CON_j^{(k-1)}} = \min\left(1, \quad \frac{CON_j^{(\ell)} \cdot f^{(\ell)}}{DIV_i^{(\ell+1)}}\right) \cdot \frac{\prod_{k=1}^{\ell+1} M_{i,j}^{(k)}}{\prod_{k=2}^{\ell+1} CON_j^{(k-1)}} \tag{22a}$$

$$= \min\left(\frac{\prod_{k=1}^{\ell+1} M_{i,j}^{(k)}}{\prod_{k=2}^{\ell+1} CON_j^{(k-1)}}, \quad \frac{M_{i,j}^{(\ell+1)}}{DIV_i^{(\ell+1)}} \cdot f^{(\ell)} \cdot \frac{\prod_{k=1}^{\ell} M_{i,j}^{(k)}}{\prod_{k=2}^{\ell} CON_j^{(k-1)}} \right) \tag{22b}$$

$$= \min\left(\frac{\prod_{k=1}^{\ell+1} M_{i,j}^{(k)}}{\prod_{k=2}^{\ell+1} CON_j^{(k-1)}}, \min_{0 \leq x \leq \ell} \left(\prod_{k=1}^{x-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(x)} \cdot \prod_{k=x+1}^{\ell+1} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \right) \right) \tag{22c}$$

$$= \min_{0 \leq x \leq \ell+1} \left(\prod_{k=1}^{x-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(x)} \cdot \prod_{k=x+1}^{\ell+1} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \right) \tag{22d}$$

In the first step we applied the definition of f from Eq. (8). To get Eq. (22b) we moved the rightmost factor into the min term, and in the following step used the induction hypothesis. The last equation follows from $\min_{1 \leq x \leq \ell+1} (g(x)) = \min\left(g(\ell+1), \min_{1 \leq x \leq \ell} (g(x))\right)$, which holds for any function g. □

D Lemmas used in the proof of optimality

We first formulate some additional lemmas and then prove Lemmas 3 and 4 used in §4.1. To simplify the notation, we drop the nodes in the paths in this section.

D.1 Auxiliary lemmas

In the following lemmas we consider an arbitrary path $\pi = [(i^1, j^1), (i^2, j^2), \dots, (i^\ell, j^\ell)]$ and denote the index for which Eq. (10) is minimized as $x^\star$.

Lemma 8. *If $x^\star \geq 3$, then the GMA allocation for the path $\widetilde{\pi} = [(\widetilde{i^2}, j^2), \dots, (i^\ell, j^\ell)]$ beginning at some interface of node 2 is still minimized at node $x^\star$.*

Proof.

$$x^\star = \arg\min_x \left(\prod_{k=1}^{x-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(x)} \cdot \prod_{k=x+1}^{\ell} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \right) \tag{23a}$$

$$= \arg\min_x \left(\prod_{k=1}^{x-1} \frac{1}{CON_j^{(k)}} \cdot \prod_{k=x+1}^{\ell} \frac{1}{DIV_i^{(k)}} \right) \tag{23b}$$

$$= \arg\min_x \left(\prod_{k=2}^{x-1} \frac{1}{CON_j^{(k)}} \cdot \prod_{k=x+1}^{\ell} \frac{1}{DIV_i^{(k)}} \right) \tag{23c}$$

$$= \arg\min_x \left(\frac{M_{\widetilde{i},j}^{(2)}}{CON_j^{(2)}} \cdot \prod_{k=3}^{x-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(x)} \cdot \prod_{k=x+1}^{\ell} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \right) \tag{23d}$$

In Eqs. (23b) and (23d) we used the fact that the allocation-matrix entries do not contribute to the argument of the minimum (they are constant among all the possible values for x). This is also true for the convergent of node 1 (because $x \geq 3$), which is used in Eq. (23c). □

Lemma 9. *If $x^\star \leq \ell - 2$, then the GMA allocation for the path $\widetilde{\pi} = [(i^1, j^1), (i^2, j^2), \dots, (i^{\ell-1}, \widetilde{j^{\ell-1}})]$ ending at some interface of node $\ell - 1$ is still minimized at node $x^\star$.*

Proof. The proof follows the same structure as the proof of Lemma 8. □

Lemma 10. *When extending the path on a non-local interface $\widetilde{i^1}$ of node 1 with some node 0 that only consists of a local and a non-local interface to $\widetilde{\pi} = [(i^0, j^0), (\widetilde{i^1}, j^1), \dots, (i^\ell, j^\ell)]$, and given that $M_{i,j}^{(0)} = CON_j^{(0)} = DIV_{\widetilde{i}}^{(1)}$, then the resulting allocation will be independent of the allocation matrix of node 0 and will still be minimized at node $x^\star$.*

Proof. We define

$$g_1(x) := \prod_{k=1}^{\ell} M_{i,j}^{(k)} \cdot \prod_{k=1}^{x-1} \frac{1}{CON_j^{(k)}} \cdot \prod_{k=x+1}^{\ell} \frac{1}{DIV_i^{(k)}}, \tag{24a}$$

$$g_0(x) := M_{i,j}^{(0)} \cdot M_{\widetilde{i},j}^{(1)} \cdot \prod_{k=2}^{\ell} M_{i,j}^{(k)} \cdot \prod_{k=0}^{x-1} \frac{1}{CON_j^{(k)}} \cdot \prod_{k=x+1}^{\ell} \frac{1}{DIV_i^{(k)}}. \tag{24b}$$

We see that

$$g_1(x^\star) = \mathcal{G}(\pi) \overset{\text{by def}}{=} \prod_{k=1}^{\ell} M_{i,j}^{(k)} \cdot \min_{1 \le x \le \ell} \left(\prod_{k=1}^{x-1} \frac{1}{CON_j^{(k)}} \cdot \prod_{k=x+1}^{\ell} \frac{1}{DIV_i^{(k)}} \right), \tag{25}$$

because $x^\star$ is the argument of the minimum of $\mathcal{G}(\pi)$. By multiplying the terms on both sides of the equation with $\frac{M_{i,j}^{(0)} \cdot M_{\widetilde{i},j}^{(1)}}{CON_j^{(0)} \cdot M_{i,j}^{(1)}}$, we get

$$g_0(x^\star) = M_{i,j}^{(0)} \cdot M_{\widetilde{i},j}^{(1)} \cdot \prod_{k=2}^{\ell} M_{i,j}^{(k)} \cdot \min_{1 \le x \le \ell} \left(\prod_{k=0}^{x-1} \frac{1}{CON_j^{(k)}} \cdot \prod_{k=x+1}^{\ell} \frac{1}{DIV_i^{(k)}} \right) \tag{26a}$$

$$= M_{i,j}^{(0)} \cdot M_{\widetilde{i},j}^{(1)} \cdot \prod_{k=2}^{\ell} M_{i,j}^{(k)} \cdot \min_{0 \le x \le \ell} \left(\prod_{k=0}^{x-1} \frac{1}{CON_j^{(k)}} \cdot \prod_{k=x+1}^{\ell} \frac{1}{DIV_i^{(k)}} \right) \tag{26b}$$

$$\overset{\text{by def}}{=} \mathcal{G}(\widetilde{\pi}). \tag{26c}$$

Equation (26b) shows that $\mathcal{G}(\widetilde{\pi})$ is still minimized at node $x^\star$; it follows from the inequality

$$g_0(x^\star) \le \frac{M_{i,j}^{(0)}}{CON_j^{(0)}} \cdot M_{\widetilde{i},j}^{(1)} \cdot \prod_{k=2}^{\ell} M_{i,j}^{(k)} \cdot \prod_{k=2}^{\ell} \frac{1}{DIV_i^{(k)}} \tag{27a}$$

$$= M_{i,j}^{(0)} \cdot M_{\widetilde{i},j}^{(1)} \cdot \prod_{k=2}^{\ell} M_{i,j}^{(k)} \cdot \prod_{k=1}^{\ell} \frac{1}{DIV_i^{(k)}}. \tag{27b}$$

Here we first used that, as $x^\star$ minimizes the right side of Eq. (26a), $g_0(x^\star)$ is at most as high as the expression of the minimum for index 1. The second step follows from $CON_j^{(0)} = DIV_{\widetilde{i}}^{(1)}$. Note that the resulting allocation $g_0(x)$ on path $\widetilde{\pi}$ is independent of node 0, because $M_{i,j}^{(0)} = CON_j^{(0)}$, meaning that those terms cancel each other out, see Eq. (24b). □

Lemma 11. *When extending the path on a non-local interface $\widetilde{j^\ell}$ of node ℓ with some node $\ell+1$ that only consists of a local and a non-local interface to $\widetilde{\pi} = [(i^1, j^1), \ldots, (i^\ell, \widetilde{j^\ell}), (i^{\ell+1}, j^{\ell+1})]$, and given that $M_{i,j}^{(\ell+1)} = CON_i^{(\ell+1)} = DIV_{\widetilde{j}}^{(\ell)}$, then the resulting allocation will be independent of the allocation matrix of node $\ell+1$ and will still be minimized at node $x^\star$.*

Proof. The proof follows the same structure as the proof of Lemma 10. □

D.2 Lemmas used in main text

Lemma 3. For $a_1, \ldots, a_x > 0$ it holds that

$$\prod_{i=1}^{x} a_i + \sum_{k=1}^{x} \left((1 - a_k) \cdot \prod_{i=k+1}^{x} a_i \right) = 1. \tag{28}$$

Proof. We do the proof by induction.

Base case ($x = 1$): $a_1 + (1 - a_1) = 1$

Inductive step:

Induction hypothesis: $\prod_{i=1}^{x} a_i + \sum_{k=1}^{x} \left((1 - a_k) \cdot \prod_{i=k+1}^{x} a_i \right) = 1.$

To show: $\prod_{i=1}^{x+1} a_i + \sum_{k=1}^{x+1} \left((1 - a_k) \cdot \prod_{i=k+1}^{x+1} a_i \right) = 1.$

Proof:

$$\prod_{i=1}^{x+1} a_i + \sum_{k=1}^{x+1} \left((1 - a_k) \cdot \prod_{i=k+1}^{x+1} a_i \right) \tag{29a}$$

$$= a_{x+1} \cdot \prod_{i=1}^{x} a_i + a_{x+1} \cdot \sum_{k=1}^{x} \left((1 - a_k) \cdot \prod_{i=k}^{x+1} a_i \right) + (1 - a_{x+1}) \tag{29b}$$

$$= a_{x+1} \cdot 1 + (1 - a_{x+1}) = 1 \tag{29c}$$

In Eq. (29c) we used the induction hypothesis. □

Lemma 4. We defined x as the index for which Eq. (10) is minimized and assume $1 < x < \ell$. We defined R_u as the sum of all allocations of all the nodes $k \in \{1, \ldots, x-1\}$ starting either at a local interface or at the local interface of some of its attached nodes, and ending either at a local interface of node u or at the local interface of some of its attached nodes, divided by $M_{i,j}^{(x)}$.

Then, it holds that

$$R_\ell = \prod_{k=x+1}^{\ell-1} b^{(k)}, \tag{30a}$$

$$R_u = \left(\prod_{k=x+1}^{u-1} b^{(k)} \right) \cdot (1 - b^{(u)}) \qquad (\text{for } x+1 \leq u \leq \ell - 1). \tag{30b}$$

Proof. Let $I^{(u)}$ be the set of interfaces of node u. For each node we will use $\perp$ to refer to its local interface. R_ℓ consists of all allocations starting at the local interface of node 1 plus all the allocations starting at the local interface of some node that is attached to one of the nodes 1 to $x - 1$, where the allocations are ending either at a local interface of node ℓ or at the local interface of some of its attached nodes:

$$R_\ell = \prod_{k=1}^{x-1} a^{(k)} \cdot \prod_{k=x+1}^{\ell-1} b^{(k)} \cdot \left(b^{(\ell)} + \sum_{\substack{t \in I^{(\ell)} \\ \setminus \{i^\ell, \perp\}}} \frac{M_{i,t}^{(\ell)}}{DIV_i^{(\ell)}} \right)$$
$$+ \sum_{1 \le p \le x-1} \sum_{\substack{t \in I^{(p)} \\ \setminus \{i^p, j^p\}}} \frac{M_{t,j}^{(p)}}{CON_j^{(p)}} \cdot \prod_{k=p+1}^{x-1} a^{(k)} \cdot \prod_{k=x+1}^{\ell-1} b^{(k)} \cdot \left(b^{(\ell)} + \sum_{\substack{t \in I^{(\ell)} \\ \setminus \{i^\ell, \perp\}}} \frac{M_{i,t}^{(\ell)}}{DIV_i^{(\ell)}} \right). \tag{31}$$

Note that for all paths going through (i^x, j^x), the argument of the minimum of Eq. (10) is always the index x: every such path can be constructed from the initial path by first dropping interface pairs at its origin and its end, and then extending the reduced path with the attached nodes. Both operations preserve x as the argument of the minimum of Eq. (10), as shown in Lemmas 8–11. Furthermore, the attached nodes do not have an influence on the GMA allocation, which is a consequence of Lemmas 10 and 11. We observed that $b^{(\ell)} + \sum_{t \in I^{(\ell)} \setminus \{i^\ell, \perp\}} \frac{M_{i,t}^{(\ell)}}{DIV_i^{(\ell)}} = 1$ and obtain

$$R_\ell = \prod_{k=1}^{x-1} a^{(k)} \cdot \prod_{k=x+1}^{\ell-1} b^{(k)} + \sum_{1 \le p \le x-1} \sum_{\substack{t \in I^{(p)} \\ \setminus \{i^p, j^p\}}} \frac{M_{t,j}^{(p)}}{CON_j^{(p)}} \cdot \prod_{k=p+1}^{x-1} a^{(k)} \cdot \prod_{k=x+1}^{\ell-1} b^{(k)} \tag{32a}$$
$$= \prod_{k=1}^{x-1} a^{(k)} \cdot \prod_{k=x+1}^{\ell-1} b^{(k)} + \sum_{1 \le p \le x-1} (1 - a^{(p)}) \cdot \prod_{k=p+1}^{x-1} a^{(k)} \cdot \prod_{k=x+1}^{\ell-1} b^{(k)} \tag{32b}$$
$$= \prod_{k=x+1}^{\ell-1} b^{(k)}, \tag{32c}$$

where we used the observation that $\sum_{t \in I^{(p)} \setminus \{i^p, j^p\}} \frac{M_{t,j}^{(p)}}{CON_j^{(p)}} = 1 - a^{(p)}$ in the step to Eq. (32b) and Lemma 3 for the last step.

With the same reasoning as above, we get, for $x + 1 \le u \le \ell - 1$,

$$R_u = \prod_{k=1}^{x-1} a^{(k)} \cdot \prod_{k=x+1}^{u-1} b^{(k)} \cdot \Bigg(\sum_{\substack{t \in I^{(u)} \\ -\{i^u, j^u\}}} \frac{M_{i,t}^{(u)}}{DIV_i^{(u)}} \Bigg)$$
$$+ \sum_{1 \le p \le x-1} \sum_{\substack{t \in I^{(p)} \\ \setminus \{i^p, j^p\}}} \frac{M_{t,j}^{(p)}}{CON_j^{(p)}} \cdot \prod_{k=2}^{x-1} a^{(k)} \cdot \prod_{k=x+1}^{u-1} b^{(k)} \cdot \Bigg(\sum_{\substack{t \in I^{(u)} \\ \setminus \{i^u, j^u\}}} \frac{M_{i,t}^{(u)}}{DIV_i^{(u)}} \Bigg) \tag{33a}$$
$$= \prod_{k=1}^{x-1} a^{(k)} \cdot \prod_{k=x+1}^{u-1} b^{(k)} \cdot (1 - b^{(u)}) \tag{33b}$$

$$+\sum_{1\leq p\leq x-1}(1-a^{(p)})\cdot\prod_{k=p+1}^{x-1}a^{(k)}\cdot\prod_{k=x+1}^{u-1}b^{(k)}\cdot(1-b^{(u)}) \tag{33c}$$

$$=\left(\prod_{k=1}^{x-1}a^{(k)}\cdot\prod_{k=x+1}^{u-1}b^{(k)}+\sum_{1\leq p\leq x-1}(1-a^{(p)})\cdot\prod_{k=p+1}^{x-1}a^{(k)}\cdot\prod_{k=x+1}^{u-1}b^{(k)}\right)\cdot(1-b^{(u)}) \tag{33d}$$

$$=\left(\prod_{k=x+1}^{u-1}b^{(k)}\right)\cdot(1-b^{(u)}). \tag{33e}$$

□

E Proofs of supplementary properties

Usability (S1). For every valid path, all the pair allocations used to calculate the allocation are positive by definition. Moreover, convergents and divergents at each node contain the respective pair allocation as part of the sum in Eq. (1), and are therefore positive. Every allocation is then positive, as it is a product of positive factors (Eq. (10)). □

Efficiency (S2). The polynomial complexity of GMA follows directly from Eqs. (8) and (9). In fact, GMA has *linear* complexity in the path length (assuming convergents and divergents are precomputed together with the allocation matrices). □

Monotonicity (S3). In the proof of monotonicity we will make use of the following lemma:

Lemma 12. *If $a, b, \delta > 0$ and $a \leq b$, then it holds that $\frac{a+\delta}{b+\delta} \geq \frac{a}{b}$.*

Proof. $\frac{a+\delta}{b+\delta} = \frac{a}{b}\cdot\frac{b\cdot(a+\delta)}{a\cdot(b+\delta)} = \frac{a}{b}\cdot\frac{ab+b\delta}{ab+a\delta} = \frac{a}{b}\cdot\left(1+\frac{\delta(b-a)}{ab+a\delta}\right) \geq \frac{a}{b}$ □

Let π be an arbitrary simple path and let node n be one of its on-path nodes. We want to show that increasing the pair allocation $M_{i,j}^{(n)}$ by some amount $\delta > 0$ does not decrease the allocation calculated by GMA for path π. Let $\mathcal{G}(\pi)$ be the formula from Eq. (10) and $x^\star$ be the argument of its minimum before increasing $M_{i,j}^{(n)}$, and let $\widehat{\mathcal{G}}(\pi)$ be the formula from Eq. (10) and $\hat{x}^\star$ the argument of its minimum after increasing $M_{i,j}^{(n)}$. We can distinguish three cases and write $\widehat{\mathcal{G}}(\pi)$ as follows:

$\hat{x}^\star < n$:

$$\widehat{\mathcal{G}}(\pi) = \prod_{k=1}^{\hat{x}^\star-1}\frac{M_{i,j}^{(k)}}{CON_j^{(k)}}\cdot M_{i,j}^{(\hat{x}^\star)}\cdot\prod_{k=\hat{x}^\star+1}^{n-1}\frac{M_{i,j}^{(k)}}{DIV_i^{(k)}}\cdot\frac{M_{i,j}^{(n)}+\delta}{DIV_i^{(n)}+\delta}\cdot\prod_{k=n+1}^{\ell}\frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \tag{34a}$$

$\hat{x}^\star = n$:

$$\widehat{\mathcal{G}}(\pi) = \prod_{k=1}^{\hat{x}^\star-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot (M_{i,j}^{(n)} + \delta) \cdot \prod_{k=\hat{x}^\star+1}^{\ell} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \tag{34b}$$

$\hat{x}^\star > n$:

$$\widehat{\mathcal{G}}(\pi) = \prod_{k=1}^{n-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot \frac{M_{i,j}^{(n)} + \delta}{CON_j^{(n)} + \delta} \cdot \prod_{k=n+1}^{\hat{x}^\star-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(\hat{x}^\star)} \cdot \prod_{k=\hat{x}^\star+1}^{\ell} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \tag{34c}$$

The following derivation holds for all of the cases above and directly proves monotonicity:

$$\widehat{\mathcal{G}}(\pi) \geq \prod_{k=1}^{\hat{x}^\star-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(\hat{x}^\star)} \cdot \prod_{k=\hat{x}^\star+1}^{\ell} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} \tag{35a}$$

$$\geq \prod_{k=1}^{x^\star-1} \frac{M_{i,j}^{(k)}}{CON_j^{(k)}} \cdot M_{i,j}^{(x^\star)} \cdot \prod_{k=x^\star+1}^{\ell} \frac{M_{i,j}^{(k)}}{DIV_i^{(k)}} = \mathcal{G}(\pi) \tag{35b}$$

To get Eq. (35a), we applied Lemma 12 to Eqs. (34a) and (34c) and the assumption that $\delta > 0$ to Eq. (34b). In the step from Eq. (35a) to Eq. (35b), we used the fact that $x^\star$ is the argument of the minimum of Eq. (10). □

F Extensibility

In real-world implementations of resource-allocation protocols, messages need to be sent on the desired paths in order to discover information about the allocation matrices of the on-path nodes. To avoid unnecessary communication overhead, we want intermediate nodes to be able to drop allocation messages if the preliminary allocation up to such a node is below a certain threshold. This is captured by the following supplementary property:

S4 Extensibility: Algorithm $\mathcal{A}$ should allow to calculate a preliminary allocation for every preliminary prefix-path π^z of length z of some terminated path π ($\pi^z = [(i^1, j^1), (i^2, j^2), \ldots, (i^z, j^z)]$ for $1 \leq z < \ell$), where we require that $\mathcal{A}(\pi^1) \geq \mathcal{A}(\pi^2) \geq \cdots \geq \mathcal{A}(\pi)$.

Theorem 13. *GMA satisfies property S4.*

Proof. For every prefix-path $\pi^z = [(i^1, j^1), (i^2, j^2), \ldots, (i^z, j^z)]$ ($2 \leq z \leq \ell$) of some terminated path π, we have

$$\mathcal{G}(\pi^z) \;=\; \left(\prod_{k=1}^{z} M_{i,j}^{(k)}\right) \cdot \min_{1 \leq x \leq z} \left(\prod_{k=1}^{x-1} \frac{1}{CON_j^{(k)}} \cdot \prod_{k=x+1}^{z} \frac{1}{DIV_i^{(k)}}\right) \tag{36a}$$

$$= \left(\prod_{k=1}^{z} M_{i,j}^{(k)}\right) \cdot \min\left(\min_{1\le x\le z-1}\left(\prod_{k=1}^{x-1}\frac{1}{CON_j^{(k)}}\cdot\prod_{k=x+1}^{z-1}\frac{1}{DIV_i^{(k)}}\right)\right.$$
$$\left.\cdot\frac{1}{DIV_i^{(z)}}, \prod_{k=1}^{z-1}\frac{1}{CON_j^{(k)}}\right) \tag{36b}$$
$$\le \left(\prod_{k=1}^{z} M_{i,j}^{(k)}\right) \cdot \min_{1\le x\le z-1}\left(\prod_{k=1}^{x-1}\frac{1}{CON_j^{(k)}}\cdot\prod_{k=x+1}^{z-1}\frac{1}{DIV_i^{(k)}}\right)\cdot\frac{1}{DIV_i^{(z)}} \tag{36c}$$
$$= \frac{M_{i,j}^{(z)}}{DIV_i^{(z)}}\cdot\mathcal{G}(\pi^{z-1}) \tag{36d}$$
$$\le \mathcal{G}(\pi^{z-1}). \tag{36e}$$

We started with Eq. (10) and in the step from Eq. (36a) to Eq. (36b) used the fact that $\min_{1\le x\le z}(f(x)) = \min\left(\min_{1\le x\le z-1}(f(x)), f(z)\right)$. The last inequality follows from Eq. (1). □

G Simulation details

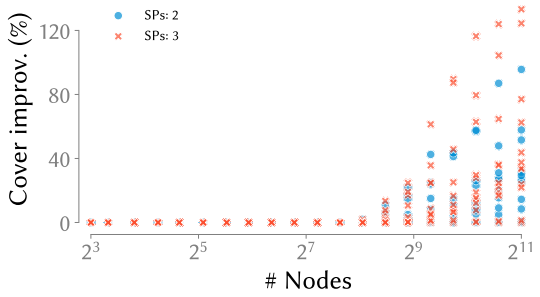


Figure 3: ***Improvement in the median 10^{-4}-cover*** *when using the 2- and 3-shortest path selection schemes instead of the single-shortest selection scheme.*

In the Barabási–Albert model, average degree and diameter are controlled by a *preferential attachment* parameter, and the total number of nodes. A higher preferential attachment will yield graphs with higher average degree and smaller diameter. We vary these two parameters to obtain 275 random graphs, with the number of nodes varying exponentially from 8 to 2048, and the attachment from 1 to 32 (the attachment always has to be smaller than the number of nodes).

The relation between the average degree and the diameter of the resulting topologies is visualized in Fig. 4. Figures 3 and 5 show additional evaluation results. Figure 6 shows the detail of the 10^{-4}-cover for each node in the graph highlighted in Figs. 2 and 5.

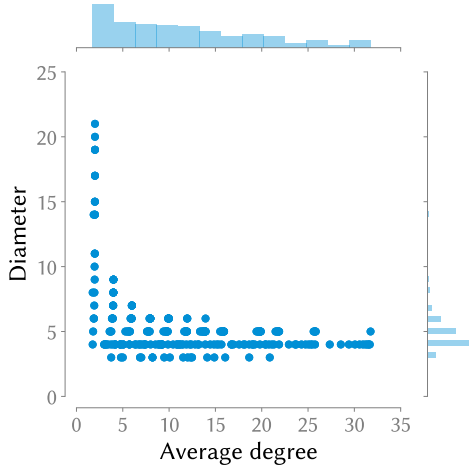


Figure 4: ***Simulated graphs by degree and diameter.*** *As the marginals show, graphs span a wide range of values in diameter and average node degree.*

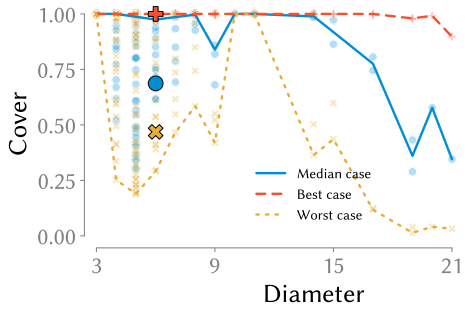


Figure 5: ***Minimum, maximum, and median single-path*** 10^{-4}***-cover breakdown.*** *The highlighted markers show the maximum* ✚*, median* ●*, and minimum* ✖ *cover for one specific graph.*

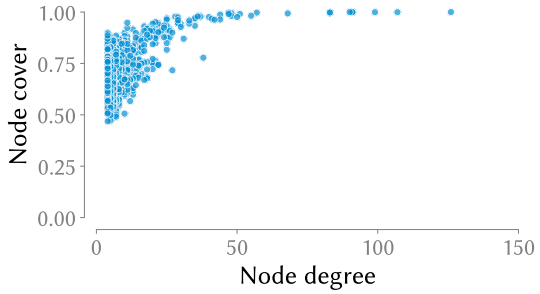


Figure 6: ***Cover and degree of a single graph.*** *Each point is a node of the graph highlighted in Figs. 2 and 5.*

References

1. Apostolaki, M., Zohar, A., Vanbever, L.: Hijacking bitcoin: Routing attacks on cryptocurrencies. In: 2017 IEEE Symposium on Security and Privacy (SP) (2017)
2. Arnold, M.: Ripple and swift slug it out over cross-border payments. https://www.ft.com/content/631af8cc-47cc-11e8-8c77-ff51caedcde6 (2018)
3. Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E.G., Song, D., Wattenhofer, R.: On scaling decentralized blockchains. In: Financial Cryptography and Data Security. Springer (2016)
4. Libra Association, T.: White paper v2.0. https://libra.org/en-US/white-paper/ (2020)
5. Standards, S.: Message definition report part 1. https://www2.swift.com/knowledgecentre/rest/v1/publications/stdsmx_pcs_mdrs/3.0/SR2020_MX_PaymentsClearingAndSettlement_MDR1_Standards.pdf?logDownload=true (2020)
6. SWIFT: SWIFT FIN traffic and figures. https://www.swift.com/about-us/swift-fin-traffic-figures/monthly-figures (2020)
7. Tradeblock: Analysis of bitcoin transaction size trends. https://tradeblock.com/bitcoin/historical/1w-f-tsize_per_avg-01101 (2015)
8. Tradeblock: Bitcoin historical data. https://tradeblock.com/bitcoin/historical/1w-f-tsize_per_avg-01101 (2020)