Hallucination Helps: Energy Efficient Virtual Circuit Routing

Antonios Antoniadis* Sungjin Im† Ravishankar Krishnaswamy‡ Benjamin Moseley§
Viswanath Nagarajan¶ Kirk Pruhs‖ Cliff Stein**

Abstract

We consider virtual circuit routing protocols, with an objective of minimizing energy, in a network of components that are speed scalable, and that may be shutdown when idle. We assume that the speed s of a link is proportional to its load, and assume the standard model for component power, namely that the power is some constant static power σ plus s^α, where typically $\alpha \in [1.1, 3]$. We give a polynomial-time offline algorithm for multicommodity routing, that has approximation ratio $O(\log^\alpha k)$, where k is the number of demand pairs. This is obtained as a combination of three natural combinatorial algorithms. The key step of the algorithm design is a random sampling technique that we call *hallucination*, which is reminiscent of the Sample-Augment framework for solving Buy-at-Bulk type problems, and sampling in cut-sparsification algorithms. The analysis of the approximation ratio is then a direct consequence of the flow-cut gap for multicommodity flow. The algorithm extends rather naturally to an online algorithm, which we show has competitive ratio $\tilde{O}(\log^{3\alpha+1} k)$. The analysis of the online algorithm introduces a natural "priority" multicommodity flow problem, and bounds the priority multicommodity flow-cut gap-this might also be of independent interest. We also explain how our hallucination technique can be used to achieve an $(O(\log km), O(\log km))$ bicriteria approximation result for the problem of buying a minimum cost collection of unit-capacitated edges to support a concurrent multicommodity flow, where m is the number of links in the network.

1 Introduction

According to the US Department of Energy [1], data networks consume more than 50 billion kWH of energy per year, and a 40% reduction in wide-area network energy is plausibly achievable if network components could dynamically adjust their speed to be proportional to demand. Virtual circuit routing, in which each connection is assigned a reserved route in the network with a guaranteed bandwidth, is used by several network protocols to achieve reliable communication [24]. In this paper we consider virtual circuit routing protocols, with an objective of minimizing energy, in a network of components that are speed scalable, and that may be shutdown when idle.

We adopt the standard models for virtual circuit routing and component energy, in particular these are the same as used in [3,4,8]. In the Energy Efficient Routing Problem (EERP), the input consists of an undirected multi-graph $G = (V, E)$, with $|V| = n$, $|E| = m$, and a collection of k request-pairs $\{(s_i, t_i) \mid s_i, t_i \in V \text{ and } i \in [k]\}$. The output is a path P_i, representing the virtual circuit for a unit bandwidth demand, between vertices s_i and t_i, for each request-pair $i \in [k]$. In the online version of the problem, the path P_i must be specified before later request-pairs become known to the algorithm. We assume that the speed of an edge is proportional to its flow, which is the number of paths that use that edge. We further assume that the power used by an edge with flow f is $\sigma + f^\alpha$ if $f > 0$, and that the edge is shutdown and consumes no power if it supports no flow. The objective is to minimize the aggregate power used over all the edges.

The term f^α is the dynamic power of the component as it varies with the speed, or equivalently load, of the component. Here $\alpha > 1$ is a parameter specifying the energy inefficiency of the components, as speeding up by a factor of s increases the energy used per unit computation/communication by a factor of $s^{\alpha-1}$. The value of α is in the range $[1.1, 3]$ for essentially all technologies [10,32]. The parameter σ is the static power, that is the power used when the component is idle, and that can only be saved by turning the component off. The static power is really only relevant/interesting if it

*University of Pittsburgh. Supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service.

†Duke University. Partially supported by NSF Award CCF-1008065.

‡Computer Science, Princeton University. Supported by the Simons Postdoctoral Fellowship.

§Toyota Technological Institute at Chicago

¶IBM T.J. Watson Research Center

‖University of Pittsburgh. Supported in part by NSF grants CCF-1115575, CNS-1253218 and an IBM Faculty Award.

**Dept. of IEOR, Columbia University. Supported in part by NSF grant CCF-0915681.

is large relative to the dynamic power of routing one unit of flow, thus we will assume that $\sigma \gg 1$.

Although speed scalable links in networks are plausible, presumably speed scaling is more likely to occur in the routers. The modeling of scalable network components by edges in [3,4,8], as well as here, is motivated primarily by reasons of mathematical tractability. Network design problems with edge costs are usually easier to solve than the corresponding problems with vertex costs. And, solutions for the vertex cost problems are often obtained by extending the solution for the edge cost problem.

1.1 The Backstory We first consider the case that the static power σ is zero. In this case, Aspens et al. [5] showed that the natural online greedy algorithm is $O(1)$-competitive. Gupta et al. [18] showed how to use convex duality to attain the same result. To understand the case when there is no static power, assume that there is a common source and common sink, that is all $s_i = s$ and all $t_i = t$, and the network consists of disjoint s-t paths. Then the convexity of the power function implies that in the optimal solution each path has the same aggregate power. The difficulty for a general network comes from the interplay between paths, but intuitively one still wants to spread the flow out over the whole network.

Let us return to the case that the static power is nonzero. If the static power is very large ($\sigma \gg k^\alpha$), then the optimal solution is essentially to route all flow over a minimum cardinality Steiner forest that connects corresponding request-pairs (since this minimizes static power); that is, the flow should be as concentrated as possible. The difficulty, in the general case, comes from the competing goals of minimizing static power, where it's best that flows are concentrated, and minimizing dynamic power, where it's best that the flows are spread out. Andrews et al. [4] showed that there is a limit to how well these competing demands can be balanced by showing that there is no polynomial-time algorithm with approximation ratio $o(\log^{1/4} n)$, under standard complexity theoretic assumptions. In contrast, Andrews et al. [3] showed that these competing forces can be "poly-log-balanced" by giving a polynomial-time poly-log-approximation algorithm. We think it is fair to say that the algorithm design and analysis in [3] are complicated and rely on big "hammers", namely the well-linked decomposition of Chekuri-Khanna-Shepherd [13], the construction of expanders via matchings of Khandekar-Rao-Vazirani [23], and edge-disjoint routings in well-connected graphs due to Rao-Zhou [28]. Moreover, the "poly" in the poly-log approximation is sufficiently large that it was not explicitly calculated in [3]. A critical parameter in [3] is $q = \sigma^{1/\alpha}$. If the flow on an edge is at least q, then one knows that the dynamic power on that edge is at least the static power, and thus static power can be charged to the dynamic power in the analysis. Roughly speaking, the algorithmic strategy in [3] is to aggregate the flow within groups, each containing q request-pairs, and then combining the above mentioned results [13,23,28] to route between groups.

Bansal et al. [8] considered the case of a common source vertex s for all request-pairs, that is all $s_i = s$. Applications for a common source vertex include data collection by base stations in a sensor network, and supporting a multicast communication using unicast routing. [8] gave a polynomial-time $O(1)$-approximation algorithm. The algorithm design and analysis is considerably easier than [3] because, after aggregation into groups, all the flow is going to the same place. [8] also gave an $O(\log^{2\alpha+1} n)$-competitive randomized online algorithm, by giving a procedure for forming the groups in an online fashion. In addition, they also provided hardness results for various generalizations of EERP.

1.2 The Story Here We present three main results in this paper, which we now discuss separately:

1. A polynomial-time $O(\log^\alpha k)$-approximation algorithm for EERP. The algorithm consists of the following two stages:

Buying Stage: The first stage of the algorithm determines which edges to use (it is convenient to say that we *buy* these edges). The algorithm first buys a Steiner forest to ensure minimal connectivity. Then each request-pair, with probability $\Theta(\frac{\log k}{q})$ *hallucinates* that it wants to route q units of flow unsplittably on a path between its end-points. Any routing algorithm that is "good" for the objective of dynamic power, for example the natural greedy algorithm from [18], is then used to route this *hallucinated flow*. All edges on which hallucinated flow is routed are then bought. Note that no actual flow is routed in this stage.

Routing Stage: The second stage of the algorithm routes the flow on the edges bought in the first stage, using any algorithm that is "good" for minimizing dynamic power.

There are two main steps in the analysis. The first step is to show by randomized rounding that the dynamic power of the hallucinated flow is comparable to OPT's total power. The second step is to show that there is a routing on the bought edges that has low dynamic power. This is accomplished by assigning each edge in the backbone a *capacity* roughly comparable to the hallucinated flow on the edge, and showing that the sparsest cut has sparsity $\Omega(\log k)$. Then by appealing to the flow-cut gap for multicommodity flows [6,25,26],

we can conclude that there must be a low-congestion routing, and hence a low dynamic power routing.

Overall, this improves on the results in [3] in the following ways: (a) the approximation ratio is better by many $\log^{\alpha} k$ factors, (b) the algorithm is much simpler, being the combination of simple combinatorial algorithms, and (c) the analysis is considerably simpler, with the only real "hammer" being the flow-cut gap for multicommodity flow. On the other hand, the results in [3] extend to the slightly more heterogeneous setting where the power used by each edge could include an edge-dependent constant multiplier. The details of our offline algorithm are in Section 3.

Hallucination is rather similar to the Sample-Augment framework [19] for solving Buy-at-Bulk type problems. This is perhaps surprising because in Buy-at-Bulk, the cost on edges is purely concave, whereas in our case the cost is convex after the jump at 0. Our algorithm analysis is quite different than the analysis of Sample-Augment for Buy-at-Bulk, and is more similar in spirit of the analysis of cut-sparsification algorithms [15,22,31].

2. A Randomized $O(\log^{3\alpha+1} k \cdot (\log\log k)^{2\alpha})$-competitive online algorithm for EERP. The offline algorithm rather naturally extends to an online algorithm: We buy the Steiner backbone edges using any of the known online algorithms for Steiner forest. Whether a request-pair should hallucinate is decided online by independent sampling. The online greedy algorithm from [18], can be used for routing hallucinated flow, and for routing the actual flow on the bought edges. The analysis however is considerably more involved than in the offline case. The analysis in [18] can be adapted to show that the dynamic power for the greedy algorithm is competitive against the power used in an optimal *priority routing*, where a request-pair can only route over edges bought by the online algorithm up until the arrival time of the request-pair. Thus to mimic the analysis in the offline case, we need to show that there is a low-congestion *priority* routing on the bought edges. This is accomplished by characterizing the notion of *sparset priority-cuts*, and then bounding the priority flow-cut gap for multicommodity flows.

We remark that this is the first poly-log-competitive online algorithm for EERP, and that we believe that our techniques for priority multicommodity flows and cuts will likely find further applications in the future. The details appear in Section 4.

3. A polynomial-time $(O(\log km), O(\log km))$ bicriteria approximation algorithm for the Multicommodity Capacitated Network Design Problem. The input to the capacitated multicommodity network design problem is a graph $G = (V, E)$ with each edge having a cost c_e and capacity q, and a collection of k unit-demand request-pairs $\{(s_i, t_i) : i \in [k]\}$ that can be supported on G. The goal is to select a minimum cost subgraph $H \subseteq G$ such that H can support a concurrent multicommodity flow of the request-pairs. [3] uses the techniques of embedding expanders via cut-matching games [23] and expander routing [28] to achieve a polynomial time algorithm that is polylog approximate with respect to the optimal cost, while supporting 1/polylog flow for each request-pair. We show in section 5 that our hallucination techniques give a polynomial time algorithm that is $O(\log km)$ approximate with respect to the optimal cost, while supporting $\Omega(1/\log km)$ flow for each request-pair.

2 Notation and Terminology

Recall that the input to EERP is an undirected multigraph $G = (V, E)$ with $|V| = n$ vertices, m edges, and k request-pairs $\{(s_i, t_i)\}_{i=1}^{k}$. The cost for routing f units of flow over any edge is: zero if $f = 0$, and $\sigma + f^{\alpha}$ if $f > 0$. We are interested in unsplitably routing request-pairs. In the analysis we will also be concerned with fractional routings. We will refer to the former as a *routing* and the latter as a *fractional routing*. The power incurred by any solution is naturally split into two parts: (i) *static power* which is σ times the number of edges with positive flow, and (ii) *dynamic power* which is $\sum_{e \in E} f_e^{\alpha}$ where f_e denotes the flow on edge $e \in E$. A useful parameter throughout the paper is $q := \sigma^{1/\alpha}$, which is the amount of flow on an edge for which the static and dynamic power are equal. We use Opt to denote the total power of a fixed optimal solution. We assume that $\alpha \geq 1$ is a constant, so any function of α is just $O(1)$.

For an undirected graph $G = (V, E)$ and subset $S \subseteq V$, we use the standard notation $\delta_G(S) := \{(u, v) \in E : u \in S, v \notin S\}$ for the *cut* corresponding to S. When the graph is clear from context, we drop the subscript. We shall sometimes refer to the vertices of these request-pairs as terminals to distinguish them from Steiner vertices in G (that do not participate in any request-pair). The *sparsity* of an edge capacitated graph G is the minimum (over all $S \subseteq V$) of the ratio of the capacity crossing cut S to the demand crossing it, i.e.

$$sparsity(G) := \min_{S \subseteq V} \frac{c(\delta(S))}{\left|\{i \in [k] : |S \cap \{s_i, t_i\}| = 1\}\right|}.$$

Here c_e is the capacity for edge e, and $c(\delta(S)) = \sum_{e \in \delta(S)} c_e$. It is well known that if the sparsity is $\Omega(\log k)$ then there is a fractional routing for the demands that respects the capacities [6,26].

In the analysis of the online algorithm, we need to define and use the notion of *priority* multicommodity flows and cuts. Here, we have an increasing sequence of unweighted multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ with respective request-pairs $\{(s_i, t_i)\}_{i=1}^{k}$.

DEFINITION 2.1. (PRIORITY MULTICOMMODITY FLOW) *Consider any sequence of multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ and request-pairs $\{(s_i, t_i) : i \in [k]\}$. A priority multicommodity flow of value λ consists of a fractional routing of λ units between s_i and t_i only using edges of multigraph $G(i)$, for each $i \in [k]$, where the total flow through any edge in this routing is at most one.*

DEFINITION 2.2. (PRIORITY CUTS) *Consider any sequence of multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ and request-pairs $\{(s_i, t_i) : i \in [k]\}$. We say that a set $Q \subseteq G(k)$ of edges priority separates pair i if and only if s_i and t_i are separated in the graph $G(i) \setminus Q$. The sparsity of a priority-cut Q is the ratio of $|Q|$ to the number of pairs that are priority separated by Q. The* **sparsest priority-cut** *is the minimum sparsity over all priority-cuts.*

DEFINITION 2.3. (PREFIX SPARSITY) *Consider any sequence of multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ and request-pairs $\{(s_i, t_i) : i \in [k]\}$. The prefix sparsity of this sequence is*

$$\min_{i=1}^{k} \quad \min_{S \subseteq V} \frac{\left|\delta_{G_i}(S)\right|}{\left|\{j \in [i] : |S \cap \{s_j, t_j\}| = 1\}\right|}.$$

For notational convenience, we use $\delta_i(S) := \left|\delta_{G_i}(S)\right|$ for any $i \in [k]$ and subset $S \subseteq V$.

Finally, we describe the online "waterfilling" algorithm from [18] that we use as a subroutine in all our algorithms. Notice that the waterfilling algorithm is concerned only with minimizing the dynamic power of the routing, and does not involve the static power σ at all. The $O(1)$ competitive analysis in [18] assumed that the graph was static, but the analysis can easily be adapted to the case that the graph is growing to show that the waterfilling algorithm is $O(1)$-competitive for priority flows.

Online Waterfilling Algorithm: The input for this algorithm is an existing multicommodity flow $\{f_e : e \in E\}$ in the graph (i.e. a routing of previous request-pairs), a new request-pair (s_i, t_i) with a demand d. The output of the algorithm is an augmentation of the existing multicommodity flow to include a flow of d units along a single $s_i - t_i$ path P_i that increases the aggregate dynamic power the least. This can be computed using a shortest path algorithm with edge costs $(f_e + d)^\alpha - f_e^\alpha$.

THEOREM 2.1. ([18]) *The waterfilling algorithm is $O(\alpha^\alpha)$-competitive for minimizing dynamic power in a static graph. The waterfilling algorithm is $O(\alpha^\alpha)$-competitive for minimizing dynamic power for a priority flow.*

3 Offline Algorithm

In this section we give a polynomial time algorithm for the Energy Efficient Routing Problem (EERP), and then show that it has approximation ratio $O(\log^\alpha k)$.

Offline Algorithm Description:
1. Constructing the Steiner backbone: Setting the cost of each edge to be the static power σ, we buy a Steiner forest that provides connectivity for the request pairs. The edges bought in this step form the Steiner backbone G_S.
2. Constructing the Hallucination backbone: Each request-pair (s_i, t_i) decides to independently "hallucinate" a demand of q, with probability $p = \min\{1, 32\lambda/q\}$. Here $\lambda = \Theta(\log k) \geq \log k$ is the flow-cut gap for multicommodity flow [6,26]. We run the waterfilling algorithm on request-pairs that hallucinate (in arbitrary order), to find an unsplittable routing $\mathcal{H}$ of q units of flow between these request-pairs. We call this routing the *hallucinated flow*. We then buy the edges in the support of $\mathcal{H}$. These edges form the hallucination backbone G_H.
3. Routing on the backbone: The waterfilling algorithm is used to route all request-pairs (with unit demand) in arbitrary order in the backbone $G_F = G_S \cup G_H$.

Note that the hallucinated flow is used solely to determine which edges to buy in the hallucination backbone, and is not a routing of actual flow.

3.1 Analysis We show that this algorithm has an approximation ratio of $O(\log^\alpha k)$ by bounding the static power used in the backbone $G_F = G_S \cup G_H$ (Steps 1 and 2) plus the dynamic power of the algorithm's routing in Step 3 as follows:
Static power of G_S: By using the Steiner forest 2-approximation algorithm from [2,17], it follows that the static power for the edges in G_S is at most twice the minimum static power required to achieve such connectivity, and hence at most $2 \cdot \mathsf{Opt}$.
Static power of G_H: The static power of the hallucination backbone G_H is at most the dynamic power of the hallucinated flow $\mathcal{H}$ since every hallucinated request-pair routes q units of flow unsplittably in $\mathcal{H}$. In Lemma 3.1 we show that the dynamic power for the hallucinated flow is $O(\lambda^\alpha) \cdot \mathsf{Opt}$ (a similar argument can be found in [8]).

Dynamic power: In order to bound the dynamic power of our routing in Step 3, we show in Lemma 3.3 that there is a routing $\mathcal{F}$ of low dynamic power in the subgraph G_F. This is sufficient, as the waterfilling algorithm used for routing within the backbone in Step 3, is $O(1)$-approximate for dynamic power [18]. To this end, we assign each edge in the backbone a *capacity* c_e equal to the amount of hallucinated flow routed on it in $\mathcal{H}$, plus $\alpha\lambda q$ if it is in the Steiner backbone G_S. Using these capacities, we show (in Lemma 3.2 and Corollary 3.1) that w.h.p, the sparsity of every cut is at least λ. Therefore, by the well-known flow-cut gap for multicommodity flow [6,25,26], there exists a *fractional* routing within the backbone G_F that respects these capacities. We then show in Lemma 3.3, by randomized rounding of this fractional multicommodity flow, that there exists an integral routing in the backbone with dynamic power $O(1)$ times the dynamic power used by the hallucinated flow $\mathcal{H}$, plus $O(\lambda^\alpha)$ times the static power used by G_S.

LEMMA 3.1. *The expected dynamic power used by the hallucinated flow $\mathcal{H}$ is $O(\lambda^\alpha) \cdot \mathsf{Opt}$.*

Proof. Let Opt denote any fixed optimal solution that for each request-pair (s_i, t_i), routes unit flow on path P_i^*. Consider any outcome of the random hallucination process. Let Opt' send q units of flow on each path P_i^* for only the hallucinated request-pairs i. We will show that the expected dynamic power used by Opt' is $O(\lambda^\alpha) \cdot \mathsf{Opt}$. Since the waterfilling algorithm is $O(1)$-approximate for the objective of dynamic power [18], it would follow that the expected dynamic power of the hallucinated flow $\mathcal{H}$ is at most $O(1)$ times the dynamic power used by Opt', i.e. $O(\lambda^\alpha) \cdot \mathsf{Opt}$.

We bound the expected dynamic power in Opt' separately for each edge $e \in E$. Fix an edge e and consider all request-pairs whose optimal paths P_i^* use e: if there are N of them, then e's power (dynamic plus static) in Opt is

$$N^\alpha + \sigma = N^\alpha + q^\alpha \geq N \cdot q^{\alpha-1}.$$

This inequality easily follows using the fact that $q^\alpha = \sigma$, and considering two cases depending on whether $N \geq q$ or not. Since each path P_i^* is chosen in Opt' independently with probability $\min\{1, 32\lambda/q\}$, we can use Corollary 6.1 with $p = \min\{1, 32\lambda/q\}$ and $D = q$ to bound the expected dynamic power for e by the following (up to an $O(1)$ factor)

$$\begin{aligned} pND^\alpha + (pND)^\alpha &\leq O(\lambda^\alpha) \cdot (Nq^{\alpha-1} + N^\alpha) \\ &\leq O(\lambda^\alpha) \cdot (2N^\alpha + \sigma) \end{aligned}$$

which is at most $O(\lambda^\alpha)$ times the power for e in Opt.

Now summing over all edges and using linearity of expectations, we conclude that the expected dynamic power in Opt' is at most $O(\lambda^\alpha) \cdot \mathsf{Opt}$. □

LEMMA 3.2. *Assume $32\lambda \leq q$. Then with probability at least $1 - k^{-3\alpha}$, the sparsity of $G_F = G_S \cup G_H$ with edge capacities $\{c_e\}$ is at least λ.*

Proof. Let $H \subseteq [k]$ denote the pairs that hallucinate in Step 2 of the algorithm. For the proof we consider a *virtual graph* $\mathcal{B}$ on vertices V with the following edges:

- *Steiner* edges: each edge $e \in G_S$ has capacity $d_e = \alpha\lambda q$ in $\mathcal{B}$.
- *Hallucinated* edges: for each $i \in H$ there is an edge (s_i, t_i) in $\mathcal{B}$ with capacity $d_{(s_i,t_i)} = q$.

Observe that for any $T \subseteq V$, the capacity of cut $\delta(T)$ in G_F is at least as much as its capacity in $\mathcal{B}$. Thus it suffices to show that the sparsity of $\mathcal{B}$ is at least λ.

Let $V_1, \ldots, V_\ell$ denote the vertices in the connected components of Steiner forest G_S. Note that these are also connected components in $\mathcal{B}$ since every request pair $\{s_i, t_i\}_{i=1}^k$ is connected in G_S. In order to lower bound the sparsity of $\mathcal{B}$, it suffices to lower bound the sparsity of each component of $\mathcal{B}$. We will show that the sparsity of any component of $\mathcal{B}$ is at least λ with probability $1 - \frac{1}{k^{4\alpha}}$. Taking a union bound over $\ell \leq k$ components in $\mathcal{B}$, would prove the lemma.

Consider any connected component of $\mathcal{B}$: note that the components in $\mathcal{B}$ are deterministic and do not depend on the random hallucination. To reduce notation, we assume in the rest of the proof that G_S is connected: so it is a Steiner tree. (Otherwise, exactly the same argument works by restricting to the request-pairs in a particular connected component.)

Since there are exponentially many cuts, we will consider cuts systematically by defining equivalence classes on cuts; then show that w.h.p. all cuts in the same class will be large compared to the demand across these cuts; and finally apply a union bound over all classes. Note that all leaves in the Steiner tree G_S are terminals. We eliminate all degree 2 Steiner vertices in G_S by short-cutting, to obtain tree G'_S. That is, every maximal path $P = \langle u, u_1, u_2, \ldots, u_\ell, v \rangle$ in G_S where $\{u_j\}_{j=1}^\ell$ are degree 2 Steiner vertices, is replaced by a single edge (u, v) in G'_S. Let $\mathcal{P}(G_S)$ denote the set of paths in G_S corresponding to edges in G'_S. Note that Steiner vertices in G'_S have degree at least three: so their number is at most the number of leaves which is at most $2k$. Thus G'_S has at most $4k$ edges.

We are now ready to define equivalence classes on cuts. We say that two cuts $C, C' \subseteq V$ are equivalent if (i) both cut exactly the same set of edges in G'_S (i.e,

the same set of paths in $\mathcal{P}(G_S)$), and (ii) both have the same set of request-pairs crossing. More precisely, $C \equiv C'$ if and only if

- $\{(u,v) \in G'_S : u \in C, v \notin C\} = \{(u,v) \in G'_S : u \in C', v \notin C'\}$, and
- $\{i \in [k] : |\{s_i,t_i\} \cap C| = 1\} = \{i \in [k] : |\{s_i,t_i\} \cap C'| = 1\}$

Let Class_j denote the set of classes that cut exactly j edges in G'_S. We first count the number of classes in Class_j: Since $|G'_S| \le 4k$, there are at most $(4k)^j$ different subsets of edges to be cut. Each cut C with $|\delta_{G'_S}(C)| = j$ divides tree G'_S into $j+1$ components. For each component, there is a choice whether/not to lie in C. So the number of classes in Class_j is at most $2^{j+1} \cdot (4k)^j \le (16k)^j$.

For each class $\mathcal{C} \in \mathsf{Class}_j$, we show that with high probability, all cuts in $\mathcal{C}$ have capacity at least λ times the demand across cuts in $\mathcal{C}$. Let $R(\mathcal{C})$ denote the set of demands across any cut in $\mathcal{C}$. We consider two cases depending on the value of $r := |R(\mathcal{C})|$.

1. $r \le \alpha jq$. In this case, since any cut in $\mathcal{C}$ has j edges from G_S (since it has j edges from G'_S), each with capacity $\alpha q\lambda$, its capacity is at least $j \cdot \alpha q\lambda \ge \lambda \cdot r$.
2. $r = |R(\mathcal{C})| > \alpha jq$. For each demand in $R(\mathcal{C})$, let X_i denote a random variable that indicates if the pair hallucinates or not. Note that $\Pr[X_i = 1] = \min\{1, 32\lambda/q\} = 32\lambda/q$; so $E[\sum_{i=1}^r X_i] = \frac{32\lambda \cdot r}{q}$. We show that the probability that a cut in $\mathcal{C}$ does not support the demand $R(\mathcal{C})$ is sufficiently small. By observing that each pair (s_i,t_i) contributes to every cut in $\mathcal{C}$ by at least qX_i, it suffices to upper-bound the following probability:

$$\begin{aligned}
\Pr\Big[q\sum_{i=1}^r X_i < r\lambda\Big] &= \Pr\Big[\sum_{i=1}^r X_i < r\lambda/q\Big] \\
&= \Pr\Big[\sum_{i=1}^r X_i < E[\sum_{i=1}^r X_i]/32\Big] \\
&\le \exp\Big(-(E[\sum_{i=1}^r X_i])/4\Big) \\
&< \exp\Big(-(\alpha jq)\cdot(32\lambda/q)/4\Big) \\
&\le \exp(-8\alpha j \log k) \quad \le \quad k^{-8\alpha j}.
\end{aligned}$$

The first inequality follows from the Chernoff bound (See Appendix 6), the second inequality uses $r > \alpha jq$, and the third inequality is by $\lambda \ge \log k$.

By summing over all $j \ge 1$ and all classes in Class_j, we derive that the probability that there is a cut with sparsity less than λ is at most

$$\begin{aligned}
\sum_{j\ge 1} |\mathsf{Class}_j| \cdot \Pr\Big[q\sum_{i=1}^r X_i < r\lambda\Big] &= \sum_{j\ge 1} (16k)^j k^{-8\alpha j} \\
&= O(1/k^{4\alpha}),
\end{aligned}$$

which completes the proof. □

COROLLARY 3.1. *With probability at least $1 - O(1/k^{3\alpha})$, there exists a capacity-respecting fractional routing of all request-pairs in backbone G_F.*

Proof. If $32\lambda > q$ then all pairs hallucinate, and the hallucinated flow $\mathcal{H}$ itself respects the capacities. If $32\lambda \le q$, then a capacity-respecting fractional routing exists w.h.p. by Lemma 3.2 and by the fact that $\lambda = \Theta(\log k)$ is the multicommodity flow-cut gap. □

LEMMA 3.3. *The expected dynamic power used by the algorithm is $O(\lambda^\alpha)$ times the static power used by G_S plus $O(1)$ times the dynamic power used by the hallucinated flow $\mathcal{H}$.*

Proof. We will bound the expected optimal dynamic power of the routing instance in Step 3, by the claimed expression. Using the $O(1)$-approximation for dynamic power from [18], this would imply the lemma.

We first consider the case where there exists a capacity-respecting fractional routing $\mathcal{G}$ in G_F. The dynamic power of $\mathcal{G}$ on any edge $e \in G_S$ with flow $g_e \le 2q\lambda\alpha$ is charged to $O(\lambda^\alpha)$ times the static power of the edges in G_S. The dynamic power for $\mathcal{G}$ on any edge e with flow $g_e > 2q\lambda\alpha$ is charged to the dynamic power of the hallucinated flow $\mathcal{H}$: g_e is at most the capacity $c_e = \alpha\lambda q + \mathcal{H}-$flow on e, so g_e is at most twice $\mathcal{H}-$flow on e.

[18] shows that a simple randomized rounding of the fractional routing $\mathcal{G}$ produces an integral routing $\mathcal{F}$ with expected dynamic power at most $O(1)$ times the dynamic power of $\mathcal{G}$ plus the edge capacities. The edge capacities are $O(\lambda)$ times the static power used by G_S plus $O(1)$ times the dynamic power used by the hallucinated flow $\mathcal{H}$.

If there is no capacity-respecting multicommodity flow, then consider the multicommodity flow that routes each (s_i,t_i) along the unique $s_i - t_i$ path in G_S. Since each edge is used at most k times, the dynamic power is at most k^α times the static power used by G_S. Corollary 3.1 states that this event can occur with probability at most $1/k^{3\alpha}$. So its contribution to the expectation is $O(1)$ times the static power of G_S. □

4 Online Algorithm

In this section we show that the offline algorithm can naturally be adapted to an online algorithm with

competitive ratio $O(\log^{3\alpha+1} k \cdot (\log\log k)^{2\alpha})$. In order to present our algorithmic ideas more transparently, we first describe the algorithm assuming that k is known in advance which we will prove is $O(\log^{3\alpha+1} k \cdot (\log\log k)^{\alpha})$-competitive. We will then show in Section 4.5 how to discharge this assumption to make the algorithm truly online at the cost of an additional factor of $O((\log\log k)^{\alpha})$ in the competitive ratio.

In response to request-pair (s_i, t_i) the algorithm takes the following steps:

1. Augmenting the Steiner backbone: We run an online algorithm for Steiner forest to connect s_i and t_i, where the cost of edges is the static power σ. Let $G_S(i)$ be the Steiner forest maintained after the i^{th} request-pair.

2. Augmenting the Hallucination backbone: The request-pair (s_i, t_i) hallucinates a demand of q with probability $p = \min(1, 32\log k/q)$. We then use the online waterfilling algorithm to find an unsplittable routing of q units of hallucinated flow between s_i and t_i. The edges used in this routing are added to the hallucinated backbone $G_H(i-1)$ to obtain $G_H(i)$.

3. Routing: We again run the online waterfilling algorithm to route one unit of actual flow between s_i and t_i in the graph $G_F(i) = G_S(i) \cup G_H(i)$, to minimize the dynamic power.

4.1 Overview of Online Algorithm Analysis We show that our online algorithm (which knows k a priori) is an $O\left(\log^{3\alpha+1} k \cdot (\log\log k)^{\alpha}\right)$-competitive for EERP as follows. For online Steiner forest, the greedy algorithm is known to be $O(\log^2 k)$-competitive [7], and there is a somewhat more complicated algorithm that is $O(\log k)$-competitive [9]. Using the latter algorithm, the static power of the edges in $G_S(k)$ is $O(\log k) \cdot \mathsf{Opt}$. As in the offline analysis, the static power for the hallucination backbone is $O(\log^{\alpha} k) \cdot \mathsf{Opt}$.

The analysis of the dynamic power of the actual multicommodity flow is more involved than in the offline case. The analysis in [18] can be adapted to show that the dynamic power of the actual multicommodity flow is $O(1)$ competitive against the dynamic power of the optimal *priority* multicommodity flow. In a priority multicommodity flow, each request-pair can only route over edges of lower priority, which are those bought by the online algorithm up until the arrival time of the request-pair. Thus to mimic the analysis in the offline case, we need to show that there is a low-congestion *priority* multicommodity flow on the backbone edges. The capacity $c_i(e)$ of each edge $e \in G_F(i)$ is the amount of hallucinated flow from the first i request-pairs that it supports, plus $(\alpha \log k) \cdot q$ if it is in the Steiner backbone $G_S(i)$. It will be convenient to view these capacities as unweighted parallel edges. For each i, we denote by $G(i)$ the multigraph consisting of $c_i(e)$ parallel edges between the end-points of each edge e. Recall the definitions of priority flow and cut, and prefix-sparsity from Section 2.

We consider the maximum priority multicommodity flow problem, fractionally priority routing as large of a fraction of each demand as possible, which can be easily expressed as a linear program. We then consider the dual linear program, in which the optimal integer solution is the sparsest priority cut. In a priority cut Q a request pair (s_i, t_i) is priority separated if s_i and t_i are separated in $G(i) \setminus Q$. Both LPs are defined formally in Section 4.3.

We show the existence of a large priority multicommodity flow using two main steps: First, in subsection 4.2 we bound the prefix-sparsity our backbone, and then use this to show that the sparsest priority-cut has sparsity $\Omega(1)$. Second, in subsection 4.3, we bound the priority flow-cut gap by $O(\log^2 k \cdot \log\log k)$. The flow-cut gap for priority multicommodity flow is the ratio between the minimum sparsity of a priority-cut and the maximum value of a priority multicommodity flow (This is analogous to the relation between concurrent multicommodity flow and sparsest cut [25,26]). Combining these results, we obtain a priority multicommodity flow of value $\Omega\left(\frac{1}{\log^2 k \cdot \log\log k}\right)$ satisfying the capacities. This implies a priority multicommodity flow of value one that satisfies capacities scaled up by $O(\log^2 k \cdot \log\log k)$, i.e. having dynamic power $O\left(\log^{3\alpha+1} k \cdot (\log\log k)^{\alpha}\right) \cdot \mathsf{Opt}$. In Section 4.4 we bring these components together to show a competitive ratio of $O\left(\log^{3\alpha+1} k \cdot (\log\log k)^{\alpha}\right)$. Then in Section 4.5 we remove the assumption that the algorithm knows k a priori with an extra factor loss of $O((\log\log k)^{\alpha})$ in competitive ratio.

4.2 Prefix Sparsity to Priority Sparsest Cut We show in Lemma 4.1 that the prefix sparsity of the backbone is $\Omega(\log k)$, and then show in Lemma 4.2 that this implies that the sparsest priority cut has value $\Omega(1)$. Notice the difference between the conditions on sparsest priority-cut and prefix-sparsity. For example, a demand j may be priority cut by some $Q \subseteq G(k)$ even though it is not separated in $G(i) \setminus Q$ for any $i > j$, i.e. j does not appear in the expression for the i^{th} prefix-sparsity. In particular, a large sparsest priority-cut clearly implies a large prefix-sparsity, but the converse (which is what we need to use) is not obvious. Lemma 4.2 proves this converse relation at the loss of a $\log k$ factor.

LEMMA 4.1. *Suppose that* $32\log k \le q$. *Then with probability of at least* $1 - O(1/k^{2\alpha})$, *the prefix sparsity of the sequence* $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ *seen in the*

online algorithm is at least $\log k$*. That is, for all* i *and all* $S \subseteq V$*, the capacity* $\delta_i(S)$ *is at least* $\log k$ *times the number of pairs* $\{(s_j, t_j)\}_{j=1}^{i}$ *crossing the cut* $(S, V \setminus S)$*.*

Proof. Fix some i. Note that multigraph $G(i)$ is an equivalent representation of the backbone $G_F(i)$ along with its capacities $c_i(\cdot)$. Since all pairs in $[i]$ have hallucinated independently, we can apply Lemma 3.2 and conclude that the sparsity of $G(i)$ is at least λ w.r.t demand pairs $[i]$. The lemma follows by a simple union bound over all $i \in [k]$. □

LEMMA 4.2. *Consider a sequence of multigraphs* $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ *with pairs* $\{(s_i, t_i) : i \in [k]\}$*. If the prefix-sparsity is at least* $\log k$*, then the sparsest priority cut is at least* $\frac{1}{3}$*.*

Proof. Consider any $Q \subseteq G(k)$ that priority separates pairs $X \subseteq [k]$. We will show that $|X| \leq 3 \cdot |Q|$, which would imply the desired lower bound on the sparsest priority-cut. Define graph $H(i) := G(i) \setminus Q$ for each $i \in [k]$. The proof is based on considering the connectivity structure in the sequence $H(1) \subseteq H(2) \subseteq \cdots \subseteq H(k)$. We say that at each *time* $i \in [k]$ the request-pair (s_i, t_i) arrives. At time j when (s_j, t_j) arrives, the edges $G(j) \setminus G(j-1) \setminus Q$ are added to graph $H(j-1)$ to get graph $H(j)$. Note that for each $i \in X$, the pair $s_i - t_i$ is separated in graph $H(i)$, by the definition of priority-cut Q.

Observe that the number of pairs in X (i.e. pairs priority cut by Q) that are disconnected in $H(k) = G(k) \setminus Q$ can be bounded by $|Q|/\log k$, since $G(k)$'s sparsest cut has value at least $\log k$. We will now upper bound the number of pairs in X that are connected in $H(k)$. For a subset of vertices $V' \subseteq V$, let $N(V') = |\{i \in X : s_i, t_i \in V'\}|$ be the total number of request-pairs in X that are induced in V'. We will show below that the sum of $N(C)$ over all components C in $H(k)$ is at most $2\,|Q|$. This would prove that $|X| \leq \frac{|Q|}{\log k} + 2|Q| \leq 3 \cdot |Q|$. In the following, we refer to the end points of request-pairs as terminals.

Toward this end we define a recurrence. Consider any $i \in [k]$ and a connected component C in $H(i)$. Let $j \leq i$ be the earliest time a request-pair arrived such that all vertices in C became connected in graph $H(j)$. Let $C_1, C_2, \ldots C_\ell$ be the components in $H(j-1)[C]$ which merged to become connected as C, at time j. By definition, $N(C_h)$ equals the number of pairs in X that are contained in C_h, for each $h \in [\ell]$. Note that $N(C)$ equals $\sum_{h=1}^{\ell} N(C_h) + I(C)$ where $I(C)$ denotes the number of pairs of X "crossing" $\{C_h\}_{h=1}^{\ell}$, i.e. pairs having end points in two distinct components among $\{C_h\}_{h=1}^{\ell}$. For each $h \in [\ell]$ define:

- $Q_h = |\delta(C_h) \cap Q|$ the number of edges in Q with exactly one endpoint in C_h, and
- $I_h = \big|\{a \in X : a \leq j-1,\ |\{s_a, t_a\} \cap C_h| = 1\}\big|$ the number of pairs in X that arrive by time $j-1$ and have exactly one end point in C_h.

We index the components $\{C_h\}_{h=1}^{\ell}$ so that C_1 contains the maximum number of terminals. We claim that $I(C) \leq \sum_{h=2}^{\ell} I_h$. To see this, note that each pair in $I(C)$ must have exactly one end-point in at least one component $\{C_h\}_{h=2}^{\ell}$. Also, since each pair $b \in I(C)$ is in X and is induced on C which gets connected at time j, we must have $b < j$: recall that for $b \in X$, s_b and t_b must be disconnected in graph $H(b)$. Thus we have

$$N(C) \quad \leq \quad \sum_{h=1}^{\ell} N(C_h) + \sum_{h=2}^{\ell} I_h.$$

We now use the prefix-sparsity condition to bound I_h. Consider the cut C_h in graph $G(j-1)$. The number of crossing edges $\big|\delta_{G(j-1)}(C_h)\big|$ is at most Q_h since C_h is a maximally connected component of $H(j-1) = G(j-1) \setminus Q$. The number of pairs crossing C_h with index at most $j-1$ is at least I_h. So the sparsity of cut C_h in graph $G(j-1)$ is bounded between:

$$\log k \leq \frac{\big|\delta_{G(j-1)}(C_h)\big|}{\big|\{a \in [j-1] : |C_h \cap \{s_a, t_a\}| = 1\}\big|} \leq \frac{Q_h}{I_h}.$$

The lower bound is by the prefix-sparisty assumption, and the upper bound is by the preceding argument. Combining the above two equations, we obtain

$$N(C) \quad \leq \quad \sum_{h=1}^{\ell} N(C_h) + \frac{1}{\log k} \cdot \sum_{h=2}^{\ell} Q_h. \tag{4.1}$$

Consider expanding this recursion to obtain $\sum_{D:comp(H(k))} N(D)$; the base case is singleton components, i.e. $N(\{v\}) = 0$ for any $v \in V$. Consider the contribution of each edge $e = (u, v) \in Q$ separately. Whenever e participates in the expression $\frac{1}{\log k} \cdot \sum_{h=2}^{\ell} Q_h$ in (4.1), the number of terminals in the component containing either u or v doubles. This is because e must have one end-point in some $\{C_h\}_{h=2}^{\ell}$ and we chose indices such that $terminals(C_1) \geq terminals(C_h)$ for all $h \in [\ell]$. Thus, the number of times e contributes is at most $2\log_2 k$, and its total contribution is $\leq \frac{2\log k}{\log k} = 2$. It follows that $\sum_{D:comp(H(k))} N(D) \leq 2 \cdot |Q|$. This completes the proof. □

4.3 Priority Flow-Cut Gap This subsection proves the following result:

THEOREM 4.1. *The flow-cut gap for priority multicommodity flow is* $O(\log^2 k \cdot \log\log k)$.

Consider any instance of priority multicommodity flow, given by a sequence $G(1) \subseteq G(2) \subseteq \cdots G(k)$ of multigraphs on vertex-set V and demand pairs $\{(s_i, t_i)\}_{i=1}^k$. We follow a natural approach by considering the LP formulation for priority multicommodity flow and its dual (which is an LP relaxation for sparsest priority-cut). We bound the flow-cut gap by showing that this sparsest priority-cut LP has a small integrality gap: this relies on a variant of the region growing approach [14,16].

The LP for priority multicommodity flow, and its dual are given below.

(PriorityFlowLP) $\max \gamma$

$$\text{s.t.} \sum_{p \in \mathcal{P}_i} f(p) \geq \gamma \qquad \forall i \in [k] \tag{4.2}$$

$$\sum_{p | e \in p} f(p) \leq 1 \qquad \forall e \in G(k) \tag{4.3}$$

$$f(p) \geq 0 \qquad \forall i \in [k], \forall p \in \mathcal{P}_i \tag{4.4}$$

(SparsestPriorityCutLP)

$$\min \sum_{e \in G(k)} d_e$$

$$\text{s.t.} \sum_{i=1}^{k} \eta_i \geq 1 \tag{4.5}$$

$$\sum_{e \in p} d_e \geq \eta_i \qquad \forall p \in \mathcal{P}_i \, \forall i \in [k] \tag{4.6}$$

$$d_e \geq 0 \qquad \forall e \in G(k) \tag{4.7}$$

$$\eta_i \geq 0 \qquad \forall i \in [k] \tag{4.8}$$

Here $\mathcal{P}_i$ denotes the set of paths from s_i to t_i in $G(i)$ and $f(p)$ denotes the flow on some path p.

The feasible solutions for the primal LP are fractional routings such that each request-pair i routes at least a γ flow between them in graph $G(i)$ (constraint (4.2)), and such that no edge supports flow more than one (constraint (4.3)). This is precisely the priority multicommodity flow problem.

In the dual, we have an LP relaxation of the *sparsest priority-cut problem*: if an integral solution Q priority-cuts k' request-pairs, we set $\eta_i = 1/k'$ for the request-pairs which are separated, and $d_e = 1/k'$ for edges in Q and 0 otherwise. The objective value is then the sparsity of the priority-cut Q.

By duality, the optimal values of these two LPs are equal. So, to prove Theorem 4.1, it suffices to upper bound the integrality gap of SparsestPriorityCutLP by $O(\log^2 k \cdot \log\log k)$. Consider any fixed optimal solution (η^*, d^*) to SparsestPriorityCutLP. First, we use a standard geometric scaling step to reduce the problem to a "priority multi-cut" problem (where the η values are "$0-1$") with an $O(\log k)$-factor loss in the sparsity. Then we apply a variant of region growing to round fractional priority multi-cut solutions to integral solutions, which loses another $O(\log k \cdot \log\log k)$ factor.

LEMMA 4.3. *For any optimal solution* (η^*, d^*) *to* SparsestPriorityCutLP*, there exists another feasible solution* (η', d') *satisfying the following properties:*

- $\sum_e d'_e \leq 8 \log k \cdot \sum_e d^*_e$*, and*
- *there is a subset* $C \subseteq [k]$ *such that* $\eta'_i = \frac{1}{|C|}$ *for* $i \in C$ *and* 0 *otherwise.*

Proof. For all $i \in [k]$ where $\eta^*_i \leq 1/(2k)$ we set $\eta^*_i = 0$. Notice that since there are at most k variables η_i, this results in a solution to SparsestPriorityCutLP where the constraint (4.5) is satisfied to extent $1/2$. We now geometrically group the η^* variables, according to classes $C_h = \{i \in [k] \mid 2^{-h} < \eta^*_i \leq 2^{-h+1}\}$ for $h \in \{1, 2, \ldots, \log(2k)\}$. Let C_ℓ be the group that maximizes $\sum_{i \in C_\ell} \eta^*_i$. Since there are at most $2 \log k$ groups and η^* totals to at least half, we have $\frac{|C_\ell|}{2^{\ell-1}} \geq \sum_{i \in C_\ell} \eta^*_i \geq \frac{1}{(4 \log k)}$.

For each $i \notin C_\ell$ set $\eta'_i = 0$ and for each $i \in C_\ell$ set $\eta'_i = \frac{1}{|C_\ell|}$. Also set d'_e to be $\frac{2^\ell}{|C_\ell|} \cdot d^*_e$ for all $e \in G(k)$. It is easy to see that (η', d') is a valid fractional solution for SparsestPriorityCutLP. Moreover, the objective $\sum_e d'_e = \frac{2^\ell}{|C_\ell|} \cdot \sum_e d^*_e \leq 8 \log k \cdot \sum_e d^*_e$. □

Fix the solution (η', d') and subset $C \subseteq [k]$ from the above lemma. By scaling d' up by a factor $|C|$, we obtain a fractional solution $\{z_e : e \in G(k)\}$ to the priority multicut instance *restricted* to the multigraph sequence $\langle G(i) : i \in C \rangle$ and pairs C. Next, we show that z can be rounded to obtain an integral solution $Q \subseteq G(k)$ that priority-cuts all the pairs in C, and has $|Q| \leq O(\log k \cdot \log\log k) \cdot \sum_e z_e$. The sparsity of such a priority-cut Q is at most:

$$\begin{aligned} \frac{|Q|}{|C|} &\leq O(\log k \cdot \log\log k) \cdot \frac{\sum_e z_e}{|C|} \\ &\leq O(\log^2 k \cdot \log\log k) \sum_e d^*_e. \end{aligned}$$

This would complete the proof of Theorem 4.1.

Bounding the integrality gap for priority multicut. Consider any instance of priority multicut given by a sequence $H(1) \subseteq H(2) \subseteq \cdots \subseteq H(r)$ of multigraphs with demand pairs $\{(s_i, t_i)\}_{i=1}^r$. The goal is to find a minimum size subset $Q \subseteq H(r)$ of edges that priority cuts each pair, i.e. $s_i - t_i$ is disconnected in $H(i) \setminus Q$ for all $i \in [r]$. Note that in our setting, $r \leq k$ the number of pairs in the sparsest priority cut instance. The natural LP relaxation for priority multicut is:

$$\min \Bigg\{ \sum_{e \in H(r)} z_e \ : \ \sum_{e \in p} z_e \geq 1 \ \ \forall p \in \mathcal{P}_i, \ \forall i \in [r], \\ z_e \geq 0, \ \forall e \in H(r) \Bigg\}.$$

Above, $\mathcal{P}_i$ is the set of $s_i - t_i$ paths in graph $H(i)$.

We first give a rounding algorithm for priority multicut that loses an $O(\log^2 r)$ factor, this is based on applying the "region growing" step from [16,25] recursively. A more careful recursion as in [14,30] can be used to obtain an $O(\log r \cdot \log\log r)$ bound. We refer to the full version of the paper for a formal description and proof.

Before the proof, we introduce some useful notation. Given a graph $L \subseteq H(r)$, let d^L denote the shortest-path metric defined by $\{z_e : e \in L\}$, i.e. $d^L(u, v)$ is the length of the shortest path between u and v with weight z on edges of L. For any vertex $v \in V$ and $\rho > 0$, define:

- $B^L(v, \rho) := \big\{u \in V : d^L(v, u) < \rho\big\}$ the *ball* of radius ρ around v in metric d^L.
- $\delta^L(v, \rho) = \{(u, w) \in L : u \in B^L(v, \rho), w \notin B^L(v, \rho)\}$ the edges cut by $B^L(v, \rho)$.
- $L(v, \rho)$ the induced graph of L on vertices $B^L(v, \rho)$.
- $\mathcal{V}^L(v, \rho) := \sum_{e \in L(v,\rho)} z_e + \sum_{(u,w) \in \delta^L(v,\rho)} \big(\rho - d^L(v, u)\big) + \frac{\mathcal{V}^*}{r} \cdot terminals\big(B^L(v, \rho)\big)$ the *volume* of ball $B^L(v, \rho)$. Here $\mathcal{V}^* = \sum_{e \in H(r)} z_e$ is the total "volume" of the LP solution.

Rounding Algorithm I. This is a careful adaptation of the LP rounding for multi-cut [16]. We first state a useful result from that paper.

LEMMA 4.4. ([16]) *For any $i \in [r]$ and $L \subseteq H(r)$ with $d^L(s_i, t_i) \geq 1$, there exists $0 < \rho < 1/2$ such that $|\delta^L(s_i, \rho)| \leq 3 \log r \cdot \mathcal{V}^L(s_i, \rho)$.*

Our rounding procedure is recursive: the input is an index $i \in [r]$ and vertex subset $U \subseteq V$ such that i is the maximum index with both $s_i, t_i \in U$. The goal is to priority-cut all pairs Π_U induced on U. (The initial call is with $i = r$ and $U = V$; the initial solution $Q = \emptyset$.) Given i and U we consider the induced graph $L = H(i)[U]$. Note that $d^L(s_i, t_i) \geq 1$ since both $s_i, t_i \in L$ and by feasibility of fractional solution z, $d^{H(i)}(s_i, t_i) \geq 1$. By applying Lemma 4.4 to both s_i and t_i, we find two radii $\rho_s, \rho_t < \frac{1}{2}$ such that:

$$|\delta^L(s_i, \rho_s)| \leq 3 \log r \cdot \mathcal{V}^L(s_i, \rho_s), \\ \text{and} \quad |\delta^L(t_i, \rho_t)| \leq 3 \log r \cdot \mathcal{V}^L(t_i, \rho_t).$$

Note that the balls $B^L(s_i, \rho_s)$ and $B^L(t_i, \rho_t)$ are disjoint. So one of them has at most $|\Pi_U|/2$ induced pairs. We consider the ball, say around s_i, which has fewer request-pairs induced inside it. We add the cut $\delta^L(s_i, \rho_s)$ to Q, and set $U_1 \leftarrow B^L(s_i, \rho_s)$ and $U_2 \leftarrow U \setminus B^L(s_i, \rho_s)$. Note that all request-pairs $j \in \Pi_U$ and having exactly one end-point in U_1 are priority-cut by Q (they are separated even in graph $H(i) \supseteq H(j)$). To handle the remaining pairs of Π_U, we recurse on U_1 (resp. U_2) with the maximum induced pair in U_1 (resp. U_2). By Lemma 4.4 the increase in $|Q|$ is at most $3 \log k$ times $\mathcal{V}^L(s_i, \rho_s)$ the volume of $H(i)[U_1]$; in this case we say that all edges in $H(i)[U_1]$ get *charged*. Moreover, by the choice of ball $B^L(s_i, \rho_s) = U_1$, the number of induced pairs in U_1 is at most $|\Pi_U|/2$ i.e. half the induced pairs in U. This implies that whenever an edge e gets charged, the number of induced pairs in the recursive call containing e reduces by a factor two: so each edge gets charged at most $\log_2 r$ times. Hence the final solution cost $|Q|$ is at most $3 \log^2 r \cdot \sum_{e \in H(r)} z_e$.

4.4 Putting the pieces together In this subsection we explain how to put the pieces together to show in Lemma 4.5 that there is witness priority multicommodity flow that routes all demands in the backbone, and whose expected dynamic power is $O(\log^{3\alpha} k \cdot (\log\log k)^\alpha)$ times the static power used by G_S plus $O(\log^{2\alpha} k \cdot (\log\log k)^\alpha)$ times the dynamic power used by the hallucinated flow $\mathcal{H}$. The proof is similar to that of Lemma 3.3.

COROLLARY 4.1. *With probability of at least $1 - O(1/k^{2\alpha})$, there exists a fractional priority routing that respects all capacities within a factor of $O(\log^2 k \cdot (\log\log k))$.*

Proof. Suppose that $32 \log k \leq q$. Then by Lemma 4.1, the prefix sparsity of the sequence of graphs $G_F(1) \subseteq G_F(2) \subseteq \ldots \subseteq G_F(k)$ is at least $\log k$ with probability at least $1 - O(1/k^{2\alpha})$. This implies that the sparsest priority-cut is at least $1/3$ by Lemma 4.2. Then the proof follows since we have shown that the priority flow-cut gap is at most $O(\log^2 k(\log\log k))$. If $32 \log k \geq q$,

all demand pairs hallucinate, and the hallucinated flow $\mathcal{H}$ is itself the desired priority multicommodity flow. □

LEMMA 4.5. *There exists an integral priority muticommodity flow that routes all demands within the backbone with expected dynamic power $O(\log^{3\alpha} k \cdot (\log\log k)^{\alpha})$ times the static power used by G_S plus $O(\log^{2\alpha} k \cdot (\log\log k)^{\alpha})$ times the dynamic power used by the hallucinated flow $\mathcal{H}$.*

Proof. Suppose that there exists a fractional priority routing that respects all capacities within factor $\gamma = O(\log^2 k \cdot (\log\log k))$; this occurs w.h.p. from Corollary 4.1. Then (as in the offline case) we use the result from [18] that randomized rounding of a fractional routing produces an integral routing, at the loss of an $O(1)$-factor in expected dynamic power. The dynamic power of flow of up to $2q\gamma \log k$ on each edge of G_S is charged to $O(\log^{3\alpha} k \cdot (\log\log k)^{\alpha})$ times the static power of the edges in G_S. The dynamic power on edges with a flow of greater than $2q\gamma \log k$ is charged to γ^{α} times the dynamic power of the hallucinated flow $\mathcal{H}$.

If there is no capacity-respecting priority multicommodity flow, consider the multicommodity flow that routes each demand (s_i, t_i) along the shortest path connecting the demand in $G_S(i)$. Since each edge is used by at most k paths, the dynamic power is at most k^{α} times the static power used by G_S. Corollary 4.1 states that this event can occur with probability at most $1/k^{2\alpha}$. So its contribution to the expectation is $O(1)$ times the static power of G_S. □

4.5 When k is not known a priori Our algorithm extends to the truly online setting when the final number k of request-pairs is not known in advance: this results in an additional $(\log\log k)^{\alpha}$ overhead in the competitive ratio. That is, we obtain a competitive ratio of $O\left(\log^{3\alpha+1} k \cdot (\log\log k)^{2\alpha}\right)$-competitive.

We refer to the full version of the paper for a formal description and proof.

5 Multicommodity Capacitated Network Design

In this section, we consider the capacitated multicommodity network design problem (CapND), as studied by [3]. In this problem, we are given a multigraph $G = (V, E)$ with each edge $e \in E$ having a cost c_e and uniform capacity q. We are also given a collection of k request-pairs $\{(s_i, t_i) : i \in [k]\}$ each with unit demand. The goal is to pick a minimum cost subgraph $H \subseteq G$ such that H can support a concurrent multicommodity flow of the request-pairs. Let $m = |E|$ denote the number of edges in G.

Andrews et al. [3] gave a (polylog, polylog) bicriteria approximation algorithm for CapND using the techniques of embedding expanders via cut-matching games [23] and expander routing [28]. That is, the cost of their solution H is polylog times the optimal cost, and H can support 1/polylog flow of each request-pair concurrently. In fact, they obtain this result for CapND as a by-product of their algorithm for EERP. In this paper, we show the following improved result for this problem.

THEOREM 5.1. *There is an $(O(\log km), O(\log km))$ bicriteria approximation for* CapND*, i.e. the solution costs $O(\log km)$ times optimal and supports $\frac{\Omega(1)}{\log km}$ flow of each request-pair.*

We note that we can also handle non-uniform demand, at the expense of an additional log factor in the approximation. A related problem that has also been studied [11,12,20] is *capacitated survivable network design*, where the requirement is for H to satisfy the flow requirement *individually for each demand* rather than concurrently. These results are incomparable to those for CapND.

We now present the details of our algorithm. First, we show that an LP-rounding algorithm gives an $(O(1), O(\log km))$ bicriteria approximation for the special case when all capacities and demands are equal to q. Then we show that the hallucination approach can be used to reduce the general case (unit demand and q capacity) to that of equal demand and capacity.

The case of equal demands and capacities. Here we consider the CapND problem when each request-pair has q units of demand (which is also the edge capacity). By scaling, we assume that all demands and capacities are one. Our algorithm is then a simple randomized rounding of the natural LP relaxation, which allows violation in capacities by $O(\log km)$.

$$
\begin{array}{lll}
(\mathsf{LP}_{\mathsf{cap}}) & \min \sum_e c_e x_e & \\
(5.9) & \text{s.t.} \sum_{p \in \mathcal{P}_i} f(p) \geq 1 & \forall i \in [k] \\
(5.10) & \sum_{p | e \in p} f(p) \leq x_e & \forall e \in E \\
(5.11) & f(p) \geq 0 & \forall i \in [k], \forall p \in \mathcal{P}_i \\
(5.12) & 0 \leq x_e \leq O(\log km) & \forall e \in E
\end{array}
$$

Here $\mathcal{P}_i$ is the set of all $s_i - t_i$ flow paths. If the LP is infeasible, then we declare that our instance is infeasible. If not, then we do a simple randomized rounding (on the flow paths for each request-pair), and

get an integral solution with expected cost equal to the LP cost, and where each edge is used by at most $O(\log km)$ request-pairs with probability $1 - \frac{1}{m}$. Thus we obtain a randomized $(O(1), O(\log km))$ bicriteria approximation algorithm.

Algorithm for CapND:
1. Constructing the Steiner backbone: Setting the cost of an edge to be c_e, we buy an (approximate) min-cost Steiner forest that provides connectivity for the request-pairs. The edges bought in this step form the Steiner backbone G_S.
2. Constructing the Hallucination backbone: Each request-pair (s_i, t_i) decides to independently "hallucinate" a demand of q, with probability $p = \min\{1, 32\lambda/q\}$. Here again $\lambda = \Theta(\log k)$ is the flow-cut gap for multicommodity flow [25,26]. Now, we use the procedure above (for uniform capacity and demands).

a. If $\mathsf{LP_{cap}}$ was infeasible then declare infeasibility for the CapND instance.

b. Else, we obtain a subgraph G_H and unsplittable routing $\mathcal{H}$ of q units between each hallucinated request-pair; $\mathcal{H}$ violates the capacities of each edge in G_H by at most $O(\log km)$. We call this routing the *hallucinated flow*.

3. Final Solution: Output the subgraph $G_F = G_S \cup G_H$.

5.1 Analysis The analysis proceeds along the same lines as in Section 3:
Cost of G_S: By using the Steiner forest approximation algorithm from [2,17], one can guarantee that cost of G_S is at most twice the minimum cost required to even achieve connectivity between the request-pairs, and hence at most $2 \cdot \mathsf{Opt}$.
Cost of G_H: Using a standard Chernoff bound, it follows that with probability $1 - \frac{1}{km}$, the demand of the hallucinated request-pairs can be routed on the support of Opt, while violating each capacity by a factor of $O(\log km)$. (This is similar to Lemma 3.1.) We now obtain a feasible LP solution to $\mathsf{LP_{cap}}$ of cost $O(\log km) \cdot \mathsf{Opt}$. Using this as a starting point, our rounding algorithm for the case of equal demands and capacities, returns a graph G_H of expected cost $O(\log km) \cdot \mathsf{Opt}$ which can unsplittably route (using $\mathcal{H}$) flow of q units for each hallucinated request-pair, while sending $O(\log km) \cdot q$ flow on each edge.
Routing on $G_S \cup G_H$: The last part of the proof shows that w.h.p. there is a flow $\mathcal{F}$ for the original request-pairs, having low congestion in the subgraph G_F. To this end, we assign "virtual capacities" $\{\hat{c}_e : e \in G_F\}$ as follows: edge $e \in G_F$ gets a virtual capacity equal to the amount of hallucinated flow routed on it in $\mathcal{H}$, plus $2\lambda \cdot q$ if it is in the Steiner backbone G_S. Using these virtual capacities, we can use Lemma 3.2 and Corollary 3.1 to conclude that with probability $1 - \frac{1}{k^3}$, the sparsity of every cut is at least λ (w.r.t the original request-pairs). Therefore, there exists a flow within the backbone G_F that respects these virtual capacities, owing to the flow-cut gap for multicommodity flow [26]. Finally, the virtual capacity of each edge in G_F is $O(\log km) \cdot q$ because $\mathcal{H}$ uses any edge to at most $O(\log km)\, q$, and we added $O(\log k) \cdot q$ units for each edge in G_S. This completes the proof of Theorem 5.1.

6 Probabilistic Inequalities

THEOREM 6.1. ([27]) *Let $X_1, X_2, ..., X_N$ be N independent random variables such that $\Pr[X_i = 0] = 1 - p_i$ and $\Pr[X_i = 1] = p_i$. Let $Y = \sum_{i=1}^{N} X_i$ and $\mu = \mathbb{E}Y$. Then for any $\delta > 0$, it follows that*

$$\Pr\left[Y \leq (1-\delta)\mu\right] \leq \exp(-\mu\delta^2/2).$$

THEOREM 6.2. ([21,29]) *Let $X_1, X_2, \ldots, X_N$ be independent non-negative random variables. Let $\alpha > 1$ and $K_\alpha = \Theta(\alpha/\log\alpha)$. Then it is the case that*

$$\left(\mathbb{E}[(\sum_i X_i)^\alpha]\right)^{1/\alpha} \leq K_\alpha \max\left(\sum_i \mathbb{E}[X_i], \left(\sum_i \mathbb{E}[X_i^\alpha]\right)^{1/\alpha}\right).$$

COROLLARY 6.1. ([8]) *Let $p \geq 0$, and let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables taking value D with probability $\min\{1, p\}$. Then $\mathbb{E}[(\sum_i X_i)^\alpha] \leq (K_\alpha)^\alpha \cdot \max\{1, pN\, D^\alpha + (pND)^\alpha\}$, where $K_\alpha = \Theta(\alpha/\log\alpha)$.*

Proof. For the case when $p \geq 1$, $X_i = D$ with probability 1, and hence we can conclude that $\mathbb{E}[(\sum_i X_i)^\alpha] = (ND)^\alpha$. For the case when $p \in [0, 1]$, $\mathbb{E}[X_i] = pD$, and $\mathbb{E}[X_i^\alpha] = pD^\alpha$. From this we can conclude that the upper bound in Theorem 6.2 is $K_\alpha \max(pND, (pN)^{1/\alpha} D)$. Taking α^{th} powers and replacing the max by a sum, we get $\mathbb{E}[(\sum_i X_i)^\alpha] \leq (K_\alpha\, D)^\alpha((pN)^\alpha + pN)$. □

References

[1] Vision and roadmap: Routing telecom and data centers toward efficient energy use, May 2009. Proceedings of Vision and Roadmap Workshop on Routing Telecom and Data Centers Toward Efficient Energy Use.

[2] Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995.

[3] Matthew Andrews, Spyridon Antonakopoulos, and Lisa Zhang. Minimum-cost network design with (dis)economies of scale. In *FOCS*, pages 585–592, 2010.

[4] Matthew Andrews, Antonio Fernández, Lisa Zhang, and Wenbo Zhao. Routing for energy minimization in the speed scaling model. In *INFOCOM*, pages 2435–2443, 2010.

[5] James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *J. ACM*, 44(3):486–504, 1997.

[6] Yonatan Aumann and Yuval Rabani. An $o(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998.

[7] Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. In *SODA*, pages 68–74, 1996.

[8] Nikhil Bansal, Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, Kirk Pruhs, and Cliff Stein. Multicast routing for energy minimization using speed scaling. In *MedAlg*, pages 37–51, 2012.

[9] Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems (extended abstract). In *STOC*, pages 344–353, 1997.

[10] David Brooks, Pradip Bose, Stanley Schuster, Hans M. Jacobson, Prabhakar Kudva, Alper Buyuktosunoglu, John-David Wellman, Victor V. Zyuban, Manish Gupta, and Peter W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.

[11] Deeparnab Chakrabarty, Chandra Chekuri, Sanjeev Khanna, and Nitish Korula. Approximability of capacitated network design. In *IPCO*, pages 78–91, 2011.

[12] Deeparnab Chakrabarty, Ravishankar Krishnaswamy, Shi Li, and Srivatsan Narayanan. Capacitated network design on undirected graphs. In *APPROX*, 2013.

[13] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *STOC*, pages 183–192, 2005.

[14] Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM*, 47(4):585–616, 2000.

[15] Wai Shing Fung, Ramesh Hariharan, Nicholas J.A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *STOC*, pages 71–80, 2011.

[16] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comput.*, 25(2):235–251, 1996.

[17] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

[18] Anupam Gupta, Ravishankar Krishnaswamy, and Kirk Pruhs. Online primal-dual for non-linear optimization with applications to speed scaling. *CoRR*, abs/1109.5931, 2011.

[19] Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximation via cost sharing: Simpler and better approximation algorithms for network design. *J. ACM*, 54(3):11, 2007.

[20] M. Hajiaghayi, R. Khandekar, G. Kortsarz, and Z. Nutov. Capacitated network design problems: hardness, approximation algorithms, and connections to group steiner tree. In *Manuscript*, 2013.

[21] William B. Johnson, Gideon Schechtman, and Joel Zinn. Best constants in moment inequalities for linear combinations of independent and exchangeable random variables. *Ann. Probab.*, (1):234–253, 1985.

[22] David R. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24(2):383–413, 1999.

[23] Rohit Khandekar, Satish Rao, and Umesh V. Vazirani. Graph partitioning using single commodity flows. *J. ACM*, 56(4), 2009.

[24] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. Addison-Wesley Publishing Company, USA, 2009.

[25] Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[26] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

[27] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[28] Satish Rao and Shuheng Zhou. Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010.

[29] Haskell P. Rosenthal. On the subspaces of L^p $(p > 2)$ spanned by sequences of independent random variables. *Israel J. Math.*, 8:273–303, 1970.

[30] Paul D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.

[31] Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011.

[32] Adam Wierman, Lachlan L. H. Andrew, and Ao Tang. Power-aware speed scaling in processor sharing systems. In *INFOCOM*, pages 2007–2015, 2009.