Hallucination Helps: Energy Efficient Virtual Circuit Routing

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

July 7, 2013

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 the router is proportional to its load, and assume the standard model for component power, namely that the power is some constant static power plus s^α, where typically $\alpha \in [1.1, 3]$. We give a polynomial-time offline algorithm that is the combination of three natural combinatorial algorithms, and show that for any fixed α the algorithm has approximation ratio $O(\log^\alpha k)$, where k is the number of demand pairs. The algorithm extends rather naturally to a randomized online algorithm, which we show has competitive ratio $\tilde{O}(\log^{3\alpha+1} k)$. This is the *first* online result for the problem. We also show that this online algorithm has competitive ratio $\tilde{O}(\log^{\alpha+1} k)$ for the case that all connections have a common source.

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 [19]. 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 [3,4,6]. In the Energy Efficient Routing Problem (EERP), the input consists of an undirected graph $G = (V, E)$, with $|V| = n$, 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

*University of Pittsburgh.
†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.
**Supported in part by NSF grant CCF-0915681.

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 [8,26]. 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 is large relative to the dynamic power of routing one unit of flow, thus we will assume that $\sigma \gg 1$.

We remark that in a real network, the scalable components (routers) would be at vertices rather than edges. The modeling of scalable network components by edges in [3,4,6] as well as here is motivated primarily by reasons of mathematical tractability. Network design problems with vertex costs tend to be harder than the corresponding edge versions. Also, obtaining algorithms for the edge version often serves as a good starting point to handle the more general vertex case.

1.1 The Backstory

We first consider the case that the static power σ is zero. In this case, [14] shows, using convex duality, that the natural online greedy algorithm is $O(\alpha)^\alpha$-competitive. 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 when flows are concentrated, and minimizing dynamic power, for which the flows are spread out. Andrews et al. [4] show 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, [3] shows that these competing forces can be balanced fairly roughly 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 [9], the construction of expanders via matchings of Khandekar-Rao-Vazirani [18], and edge-disjoint routings in well-connected graphs due to Rao-Zhou [23]. Moreover, The "poly" in the poly-log approximation is fairly 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 terminals, and then cleverly combining the above mentioned results [9,18,23] to route between groups.

The work [6] 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. [6] gives a polynomial-time $O(\alpha)$-

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. Bansal et al. [6] also gives an $O(\log^{2\alpha+1} k)$-competitive randomized online algorithm, by giving a procedure for forming the groups in an online fashion. In addition, they also 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 two parts, described below.

Buying Stage: The first part 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 (using just the static power of σ) 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. But now, when the request-paired demands are all at least q, the static power costs are dominated by the dynamic power costs. So we can simply use the natural greedy algorithm for minimizing dynamic power from [14], to solve this *hallucinated flow problem*. We use this to decide which edges to *buy*: those on which hallucinated flow is routed. Notice that the first stage simply chooses which edges to buy (i.e., pay the static power) and does not actually route any flow yet.

Routing Stage: The second part of the algorithm simply routes the flow on the edges bought in the first step, using the natural greedy algorithm from [14] to minimize the dynamic power, ignoring the static power of σ since we've already bought these edges in the first phase.

The two main steps in the analysis are: (i) showing that unsplittably routing q units of flow between the hallucinated demands has dynamic power comparable to OPT's total cost (static plus dynamic power), and (ii) showing that there is a routing on the edges bought which has low dynamic power. While the first is a simple randomized rounding step, we establish the second by assigning *virtual capacities* on edges equal to the flow routed by the hallucinated routing in the first stage, and then showing that the sparsest cut in the graph with virtual capacities (and demands of the original request-pairs) is large, namely $\Omega(\log k)$. By appealing to the flow-cut gap for multicommodity flows [20,21]), we can get a low-congestion routing respecting the virtual capacities, and consequently, bound the dynamic power.

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. The details are in Section 3.

We note that our framework is rather similar to the Sample-Augment framework [15] for solving Buy-at-Bulk type problems. This is surprising because in these problems, the cost on edges is purely concave, whereas in our case it is convex after the jump at 0. From a technical stand point, the proof ideas are very different from the Buy-at-Bulk works, and in this aspect, the hallucination step is more similar in spirit to cut-sparsification algorithms for undirected graphs [11,17,25].

2. 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 greedy algorithm that is used for buying

the hallucinated backbone edges is already online [14]. Finally, we can also route the demands on the edges bought (to minimize dynamic power) using the online algorithm from [14].
The analysis however is considerably more involved than in the offline case. Since the backbone graph on which the online routing algorithm in the routing stage operates is itself bought online, our dynamic power is only competitive against "*priority routings*" where a request-pair can only route over edges bought by the online buying algorithm up until the arrival time of the request-pair. (The priority routings, however can use the knowledge of the entire graph bought after all request-pairs arrive.) To this end, we show that there is a low-congestion priority multicommodity flow on the bought edges (w.r.t the virtual capacities), by (i) characterizing the notion of *sparsest priority-cuts*, and then (ii) bounding the flow-cut gap for priority multicommodity flows.
We remark that this is the *first* poly-log-competitive online algorithm for EERP, and believe that our techniques for priority flows and cuts will likely find further applications in the future. The details appear in Section 4.

3. Our online algorithm is $O(\log^{\alpha+1} k)$-competitive for single source EERP. This improves on the result in [6] in the following ways: (a) the competitive ratio is better by a factor of $O(\log^{\alpha} k)$, (b) the algorithm is simpler as it avoids the use of exponential weighting functions, and (c) the analysis is somewhat simpler. The analysis is easier than for the analysis of our online algorithm in Section 4 for the multiple-source-sink case because we are able to adopt a simple technique from [6] to show that there is a low congestion priority flow. We cover this in Section 5.

2 Notation and Terminology

Recall that the input to EERP is an undirected graph $G = (V, E)$ with $|V| = n$ vertices, 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 $f > 0$. 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.
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.
Finally, we describe the online "waterfilling" algorithm from [14], that we use as a subroutine in all our algorithms.

Online Waterfilling Algorithm $\mathcal{WA}$ Description: The input for this algorithm is an existing 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 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 (which can be computed using a shortest path algorithm).

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.

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 $O(\log^{\alpha} k)$.

3.1 Algorithm Description

We are now ready to present our offline algorithm.

Offline Algorithm Description:

1. Constructing the Steiner backbone: Setting the cost of an edge to be the static power σ, we first buy an (approximate) minimum-cost Steiner forest [2,13]. We denote the set of edges bought in this step as 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 [21]. We run the algorithm $\mathcal{WA}$ (in an arbitrary order of the hallucinated demands) to find an unsplittable routing $\mathcal{H}$ of q units of flow between the end points of each hallucinated request-pair. We call this routing the *hallucinated flow*. We then buy each edge of $\mathcal{H}$, i.e., pay the static power of σ, and include it the hallucination backbone G_H. Note that no flow is actually routed in this step, and the waterfilling algorithm is used only to determine which edges to buy in the hallucination backbone.

3. Routing on the backbone: We then again run the waterfilling algorithm to route all request-pairs (with unit demand) in the graph $G_F = G_S \cup G_H$, to minimize the dynamic power.

3.2 Analysis

We need to bound 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 above. This proceeds along the following lines.

Static power of G_S: By using the Steiner forest approximation algorithm from [2,13], one can guarantee 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 the every hallucinated request-pair routes q units of flow unsplittably in $\mathcal{H}$. Lemma 1 shows that the dynamic power for the hallucinated flow is $O(\lambda^{\alpha}) \cdot \mathsf{Opt}$ (a similar argument can be found in [6]).

Dynamic power: In order to bound the dynamic power of our routing in Step 3, we show in Lemma 4 that there is a good fractional 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 $O_{\alpha}(1)$-approximate for dynamic power [14], even against fractional solutions.

To this end, we assign *virtual capacities* $\{c_e : e \in E\}$ to each edge in the backbone G_F 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 virtual capacities, we show (in Lemma 2) that w.h.p, the sparsity of every cut is at least λ (w.r.t the original demands of the request-pairs). Therefore, there exists a *fractional* routing within the backbone G_F that respects these virtual capacities, owing to

the flow-cut gap for multicommodity flow [20]. Let $\mathcal{F}$ be such a flow. Using the fact that we assign the virtual capacities based the hallucinated flow $\mathcal{H}$ (plus an extra capacity of $\lambda\alpha q$ for the Steiner edges G_S), we can show in Lemma 4 that the aggregate power used by the flow $\mathcal{F}$ is at most $O_\alpha(1)$ times the dynamic power used by the hallucinated flow $\mathcal{H}$, plus $O(\lambda^\alpha)$ times the static power used by G_S. Finally, consider the case that here is no fractional flow within G_F that satisfies the virtual capacities which occurs with probability at most $1/k^{4\alpha}$. In this case let $\mathcal{F}$ be the most obvious witness flow that only uses the edges in G_S. Since the congestion on each edge is at most k, the dynamic power used by $\mathcal{F}$ is at most k^α times the static power used by G_S. Then this high power is offset by the low probability $1/k^{4\alpha}$.

We now present the details.

Lemma 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 [14], 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 (dynamic plus static) power 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 into Opt' independently with probability $\min\{1, 32\lambda/q\}$, we can use Corollary 25 with $p = \min\{1, 32\lambda/q\}$ and $D = q$ to bound the expected dynamic power for e by the following (upto an $O(1)$ factor)

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

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 $O(\lambda^\alpha) \cdot \mathsf{Opt}$. □

We now move on to bounding the dynamic power of the routing in Step 3. To this end, let us assign a *virtual capacities* c_e on edge e equal to the flow $\mathcal{H}$ routes on e plus $\alpha\lambda q$ if $e \in G_S$. Now consider the multicommodity flow instance where there is unit demand for each request-pair $\{(s_i, t_i)\}_{i=1}^k$. The *sparsity* of G_F is the minimum (over all $S \subseteq V$) of the ratio of the capacity crossing cut S to the demand crossing it, i.e.

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

Above $c(\delta(S)) = \sum_{e \in \delta(S)} c_e$. 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 [20,21]. The crux of our proof is then the following lemma, which shows that the backbone G_F has high sparsity.

Lemma 2 *Assume $32\lambda \leq q$. Consider any κ such that $\lambda \geq \log \kappa \geq \log k$. Then with probability at least $1 - \kappa^{-4\alpha}$, the sparsity of $G_F = G_S \cup G_H$ is at least λ.*

Proof: Since there are exponentially many cuts, we will consider cuts systematically by defining equivalence classes on cuts, and then show that all cuts in the same class will be large compared to the demand across these cuts. To this end, we first eliminate degree 2 vertices in G_S (only for analysis): in G_S, replace every minimal path P between u and v of degree greater than 2 by an edge (u, v). Let $\mathcal{P}(G_S)$ denote the set of paths in G_S corresponding to edges in G'_S. Note that any leaf node of G'_S belongs to some request-pair since the original graph G_S has the same property. Furthermore, G'_S has at most $4k$ edges by observing that there are at most $2k$ terminals, and the number of internal nodes is at most the number of leaf nodes.
We are now ready to define equivalence classes on cuts. We say that two cuts C and C' are equivalent if both cut exactly the same set of edges in G'_S (i.e, the same set of paths in $\mathcal{P}(G_S)$), and have the same set of request-pairs crossing. More precisely, $C \equiv C'$ if and only if

- $C \cap G'_S = C' \cap G'_S$, and
- A request-pair (s_i, t_i) is separated by C if and only if it is also separated by C'.

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 $|G'_S| \leq 4k$, there are at most $(4k)^j$ different subsets of edges to be cut. Furthermore, each cut C and C' divides G'_S into $j+1$ components by deleting j edges. For each component, there is a choice of which side of the cut to belong. We can then conclude that the number of classes in Class_j is at most $2^{j+1} \cdot (4k)^j \leq (16k)^j$.
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 cuts in $\mathcal{C}$. We consider two cases. Firstly, consider the case where $r := |R(\mathcal{C})| \leq \alpha j q$. In this case, since all cuts in $\mathcal{C}$ have j edges from G_S (since it has j edges from G'_S), each with capacity $\alpha q \lambda$, we are done. Secondly, consider the case where $r := |R(\mathcal{C})| > \alpha j q$. For each demand in $R(\mathcal{C})$, let X_i denote a random variable that indicates if the pair hallucinates or not. Note that $\Pr[X_i = 1] = \min\{1, 32\lambda/q\} = 32\lambda/q$. 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:

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

The third inequality follows from the Chernoff bound (See Appendix A). By summing over all $j \geq 1$ and all classes in Class_j, we derive that the probability that there is a cut C of G_F whose demand across C is more than the capacity of C divided by λ is at most

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

which completes the proof. □

Corollary 3 *With probability at least $1 - O(1/k^{4\alpha})$, there exists a multicommodity flow $\mathcal{F}$ sending one unit flow between the request-pairs while respecting the virtual capacities on the backbone G_F.*

Proof: If $32\lambda > q$ then all pairs hallucinate, and the hallucinated flow $\mathcal{H}$ itself respects the virtual capacities. If $32\lambda \leq q$, the statement follows by Lemma 2 with $\kappa = k$ and by the fact that $\lambda = \Theta(\log k)$ is the multicommodity flow-cut gap. □

Lemma 4 *There exists a fractional routing $\mathcal{F}$ of all demands within G_F whose expected dynamic power is $O(\lambda^\alpha)$ times the static power used by G_S plus $O_\alpha(1)$ times the dynamic power used by the hallucinated flow $\mathcal{H}$.*

Proof: We first consider the case where there exists a capacity-respecting flow using the virtual capacities on G_F. In this case, we let $\mathcal{F}$ be such a flow. The dynamic power of $\mathcal{F}$ for a flow of up to $2q\lambda\alpha$ on each edge of G_S is charged to $O(\alpha^\alpha\lambda^\alpha)$ times the static power of the edges in G_S. The dynamic power for $\mathcal{F}$ on edges with a flow of greater than $2q\lambda\alpha$ is charged to the dynamic power of the hallucinated flow $\mathcal{H}$ (which is at least $q\lambda\alpha$ since the Steiner capacity is at most $q\lambda\alpha$ on any edge), upto an additional factor 2^α. If there is no capacity-respecting flow, then let $\mathcal{F}$ be the 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 used by $\mathcal{F}$ is at most k^α times the static power used by G_S. Corollary 3 states that this event can occur with probability at most $1/k^{4\alpha}$, hence we derive the lemma. □

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 4.4 how to discharge this assumption to make the algorithm truly online with an additional factor of $O((\log\log k)^\alpha)$.

Our Online Algorithm Description: 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 $\mathcal{WA}$ to find an unsplittable routing of q units of 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 algorithm $\mathcal{WA}$ (in a separate instance) to route unit demand between s_i and t_i in the graph $G_F(i) = G_S(i) \cup G_H(i)$, to minimize the dynamic power.

Theorem 5 *Suppose the number k of demands is known prior to the algorithm. Then there is an $O\left(\log^{3\alpha+1} k \cdot (\log\log k)^{\alpha}\right)$-competitive randomized online algorithm for EERP.*

The analysis of our online algorithm proceeds along the following lines. For online Steiner forest, the greedy algorithm is known to be $O(\log^2 k)$-competitive [5], and there is a better $O(\log k)$-competitive algorithm [7]. Using this, the static power of the edges in $G_S(k)$ is $O(\log k) \cdot \mathsf{Opt}$. We buy the hallucinated backbone $G_H(k)$ as in the offline case, and the total static power is again $O(\log^\alpha k) \cdot \mathsf{Opt}$. We are then left to bound the dynamic power used by our online algorithm which routes greedily on the bought edges in Step 3.
The analysis here is more involved than in the offline case. Indeed, since we run the online routing algorithm in Step 3 on a backbone graph that is changing (where earlier terminals are not given access to edges bought later to route their flow), the waterfilling algorithm from [14] is only competitive against an offline routing which also respects these priorities w.r.t when edges are bought. We will show that there exists a *priority multicommodity flow* on the backbone $G_F(k)$ that has dynamic power at most $O\left(\log^{3\alpha+1} k \cdot (\log\log k)^\alpha\right) \cdot \mathsf{Opt}$: in a priority flow, each request-pair (s_i, t_i) is routed only on $G_F(i)$, i.e., using edges bought no later than the time that the request-pair arrived. Since the waterfilling algorithm is α^α-competitive [14] against such offline solutions, this would bound our online algorithm's dynamic power by $O\left(\log^{3\alpha+1} k \cdot (\log\log k)^\alpha\right) \cdot \mathsf{Opt}$, and prove Theorem 5.

Existence of Priority Multicommodity Flow. As in the offline case, we first add *virtual capacities* $c_i : E \to \mathbf{Z}^+$ to the edges bought: more specifically, for all i, the virtual capacity of each edge $c_i(e)$ 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 $G_S(i)$. Observe that the virtual capacities are non-decreasing with arrivals, i.e., $c_i(e) \geq c_{i-1}(e)$. It will be convenient to view these virtual 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. We will show the existence of a large priority multicommodity flow on this sequence $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ of multigraphs.

Definition 6 (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.*

The maximum priority multicommodity flow can be easily expressed as a linear program. To show a large priority flow, we consider the dual to this LP formulation. (Both LPs are defined formally in Section 4.2.) Integral solutions to the dual will correspond to *priority-cuts*, defined as follows.

Definition 7 (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.*

We show the existence of a large priority multicommodity flow using two main steps: (i) bounding the priority "flow-cut" gap, and (ii) proving a large sparsest priority-cut value. 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 [20,21]). We show that the flow-cut gap for any priority multicommodity flow instance is $O(\log^2 k \cdot \log\log k)$. Then we show that the sparsest priority-cut of the sequence $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ resulting from our algorithm is large, with high probability. The key property used in showing this is a condition similar to Lemma 2 (in the offline algorithm) which says that *for each i*, the sparsest cut in the graph $G(i)$ (restricted to the first i demand pairs) is large. Formally,

Definition 8 (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 $S \subseteq V$.

Lemma 9 *Suppose that $32 \log k \leq q$. 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 virtual capacities $c_i(\cdot)$. Since all pairs in $[i]$ have hallucinated independently, we can apply Lemma 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 $i \in [k]$. □

Summary: In the next subsection we use Lemma 9 to show that the sparsest priority-cut is $\Omega(1)$. Then in the following subsection, we bound the priority flow-cut gap by $O(\log^2 k \cdot \log\log k)$. 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 virtual capacities. This implies a priority multicommodity flow of value one that satisfies virtual capacities scaled up by $O(\log^2 k \cdot \log\log k)$, i.e. having dynamic power $O\left(\log^{3\alpha+1} k \cdot (\log\log k)^{\alpha}\right) \cdot \mathsf{Opt}$. The actual proof involves some case analyses depending on the prefix sparsity value and in Section 4.3 we give explain how we put things together in detail to prove Theorem 5. Then in Section 4.4 we remove the assumption that the algorithm knowns k prior with an extra factor loss of $O((\log\log k)^{\alpha})$ in competitive ratio.

4.1 Prefix Sparsity to Priority Sparsest Cut

Here, we prove an important relation between the sparsest priority-cut and prefix sparsity.

Lemma 10 *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 $\frac{1}{3}$.*

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 is not obvious. Lemma 10 proves this converse relation at the loss of a $\log k$ factor.

Proof of [Lemma 10] 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 $N(V') = |\{j \in X : s_i, t_i \in V'\}|$ 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 point of a request-pair as a terminal.

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 some 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 connected component of $H(j-1) = G(j-1) \setminus Q$. The number of crossing pairs 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:

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

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

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

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 (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. □

4.2 Priority Flow-Cut Gap

This subsection proves the following result:

Theorem 11 *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 []. We refer to edges in $G(i) \setminus G(i-1)$ as having *priority* i. The LP for priority multicommodity flow, and its dual are given below.

$$\max \gamma \qquad \text{(PriorityFlowLP)}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}_i} f(p) \geq \gamma \qquad \forall i \in [k] \quad (2)$$

$$\sum_{p | e \in p} f(p) \leq 1 \qquad \forall e \in G(k) \quad (3)$$

$$0 \leq f(p) \leq 1 \qquad \forall i \in [k], \forall p \in \mathcal{P}_i \quad (4)$$

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

$$\text{s.t.} \sum_{i=1}^{k} \eta_i \geq 1 \qquad (5)$$

$$\sum_{e \in p} d_e \geq \eta_i \qquad \forall p \in \mathcal{P}_i \, \forall i \in [k] \qquad (6)$$

$$d_e \geq 0 \qquad \forall e \in G(k) \qquad (7)$$

$$\eta_i \geq 0 \qquad \forall i \in [k] \qquad (8)$$

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 (2)), and such that no edge supports flow more than one (constraint (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' terminals, we set $\eta_i = 1/k'$ for the terminals 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 11, it suffices to upper bound the integrality gap of SparsestPriorityCutLP by $O(\log^2 k \cdot \log\log k)$. Consider any fixed optimal solution to SparsestPriorityCutLP (η^*, d^*). First, we use a standard geometric scaling step to reduce the problem to a "priority multi-cut" problem, where the η values are in $\{0, 1\}$, 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 12 *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 (5) is satisfied to extent $1/2$. We now geometrically group the η^* variables, according to classes $C_h = \{i \in [k] \mid 2^{-h} <$

$\eta_i^* \le 2^{-h+1}\}$ 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 $\frac{|C_\ell|}{2^{\ell-1}} \ge \sum_{i \in C_\ell} \eta_i^* \ge \frac{1}{(4 \log k)}$.
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 $\sum_e d'_e = \frac{2^\ell}{|C_\ell|} \cdot \sum_e d_e^* \le 8 \log k \cdot \sum_e d_e^*$. □
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 $|Q| \le O(\log k \cdot \log\log k) \cdot \sum_e z_e$. The sparsity of such a priority-cut Q is at most:

$$\frac{|Q|}{|C|} \quad \le \quad O(\log k \cdot \log\log k) \cdot \frac{\sum_e z_e}{|C|} \quad \le \quad O(\log^2 k \cdot \log\log k) \sum_e d_e^*.$$

This would complete the proof of Theorem 11.

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 $i \in [r]$. The natural LP relaxation for this problem is:

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

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 "region growing" [] recursively. Then we show that a more careful recursion as in [] can be used to obtain an $O(\log r \cdot \log\log r)$ bound.
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:

- $B^L(v, \rho) := \{u \in V : d^L(v, u) < \rho\}$ 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 $B^L(v, \rho)$.
- $\mathcal{V}^L(v, \rho) := \sum_{e \in L(v,\rho)} z_e + \sum_{(u,w) \in \delta^L(v,\rho)} \left(\rho - d^L(v, u)\right) + \frac{\mathcal{V}^*}{k} \cdot terms\left(B^L(v, \rho)\right)$ 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 [12]. We first state a useful result from that paper.

Lemma 13 ([12]) *For any $i \in [r]$ and $L \subseteq H(r)$ with $d^L(s_i, t_i) \ge 1$, there exists $0 < \rho < 1/2$ such that $|\delta^L(s_i, \rho)| \le 3 \log k \cdot \mathcal{V}^L(s_i, \rho)$.*

Our rounding procedure is recursive: the input is an index $i \in [k]$ 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 $i = k$ 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 13 to both s_i and t_i, we find two radii $\rho_s, \rho_t < \frac{1}{2}$ such that:

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

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 in Π_U and having exactly one end-point in U_1 are priority-cut by Q (they are separated even in graph $H(i)$). 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 13 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 reduced by a factor two: so each edge gets charged at most $\log_2 k$ times. Hence the final solution cost $|Q|$ is at most $3 \log^2 k \cdot \sum_{e \in H(r)} z_e$.

Rounding Algorithm II. Here we make use of a different region growing result:

Lemma 14 ([10]) *For any $i \in [r]$ and $L \subseteq H(r)$ with $d^L(s_i, t_i) \geq 1$, there exists $0 < \rho < 1/2$ s.t.*

$$|\delta^L(s_i, \rho)| \quad \leq \quad 4 \cdot \mathcal{V}^L(s_i, \rho) \cdot \log \left(\frac{\mathcal{V}^L(s_i, 1/2)}{\mathcal{V}^L(s_i, \rho)} \right) \cdot \log \log \left(\frac{\mathcal{V}^L(s_i, 1/2)}{\mathcal{V}^*/k} \right).$$

Vish: complete this.

4.3 Putting the pieces together

In this section we show how we prove Theorem 5 by putting what we proved together.

Corollary 15 *Consider the random priority multicommodity flow instance $\mathcal{M}$ seen in Step 3 of the online algorithm: sequence $G(1) \subseteq G(2) \subseteq \cdots G(k)$ of multigraphs with request-pairs $\{(s_i, t_i)\}_{i=1}^k$. Then with probability of at least $1 - O(1/k^{2\alpha})$, there exists a priority multicommodity flow 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 9, the prefix sparsity of the sequence of graphs $G_F(1) \subseteq G_F(2) \subseteq ... \subseteq G_F(k)$ is at least $\log k$ with probability at least $1 - O(1/k^{2\alpha})$. This implies that the sparest priority-cut is at least $1/3$ by Lemma 10. Then the proof follows since we have shown that the prefix 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 the desired priority multicommodity flow. □

As discussed before, the static power used by the algorithm can be easily upper bounded by $O(\log^\alpha k)\mathsf{Opt}$ as in the offline case. Further, it suffices to show that there exists a witness priority multicommodity flow $\mathcal{F}$ of low dynamic power in expectation since the waterfilling algorithm yields a routing of dynamic power $O_\alpha(1)$ times the dynamic power used by a witness priority flow. This is formally stated in the following lemma, and the proof is very similar to that of Lemma 4.

Lemma 16 *There exists a priority muticommodity flow $\mathcal{F}$ that route all demands within $\{G_F(i)\}_{i\in[k]}$ whose expected dynamic power is $O(\log^{3\alpha} k \cdot (\log\log k))$ times the static power used by G_S plus $O(\log^{2\alpha} k \cdot (\log\log k))$ times the dynamic power used by the hallucinated flow $\mathcal{H}$.*

Proof: Suppose that there exists a priority muticommodity flow that respect all capacities within factor $\gamma = O(\log^2 k \cdot (\log\log k))$. Then we let $\mathcal{F}$ be such a priority flow. The dynamic power of $\mathcal{F}$ for a flow of up to $q\gamma \log k$ 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 for $\mathcal{F}$ on edges with a flow of greater than $q\gamma\lambda$ is charged to γ^α times the dynamic power of the hallucinated flow $\mathcal{H}$ (with a loss of factor 2^α). If there is no capacity-respecting priority flow, we let $\mathcal{F}$ be the 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 times, the dynamic power used by $\mathcal{F}$ is at most k^α times the static power used by G_S. Corollary 15 states that this event can occur with probability at most $1/k^{4\alpha}$, hence the lemma follows. □
Due to the facts that the static power of G_S is $O(\log^\alpha k)\mathsf{Opt}$ and the dynamic power used by the hallucinated flow $\mathcal{H}$ is $O(\log^\alpha k)\mathsf{Opt}$ in expectation (by a straightforward modification of Lemma 1), we derive Theorem 5.

4.4 When k is not known *a priori*

Our algorithm extends easily 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 (cf. Theorem 5).

Theorem 17 *There is an $O\left(\log^{3\alpha+1} k \cdot (\log\log k)^{2\alpha}\right)$-competitive randomized online algorithm for EERP.*

When request-pair i arrives (in Step 2 of the algorithm), we hallucinate a demand q with probability $\min\left(1, (32\log i)\cdot\frac{1}{q}\right)$. In addition, we maintain the invariant that by the arrival time of the i^{th} request-pair, *every* pair $j \in \{1, 2, \ldots, i\}$ hallucinates with probability $\approx \min\left(1, (32\log i)\cdot\frac{1}{q}\right)$. This can be ensured by sampling each request-pair j several times, each with probability $\frac{1}{q}$: in particular, at the arrival time of request-pair j it is sampled $\lceil 64\log j\rceil$ times, and after each subsequent request-pair $i > j$ we sample j more times to ensure a total of $\lceil 64\log i\rceil$ samples. This results in the following (weaker) version of Lemma 9, with an identical proof.

Lemma 18 *For each $i \in [k]$, with probability at least $1 - O(1/i^{2\alpha})$, the sparsity of the multigraph $G(i)$ with demands $\{(s_j, t_j)\}_{j=1}^i$ is at least $\log i$.*

Using this condition (instead of the $\log k$ prefix-sparsity from Lemma 9) we can prove (see Lemma 19 below) that the sparsest priority-cut is $\Omega(1/\log\log k)$ (instead of constant). Combined with the flow-cut gap (Theorem 11) we would obtain a priority multicommodity flow of value $\frac{\Omega(1)}{(\log k\cdot\log\log k)^2}$ satisfying the virtual capacities. This would imply Theorem 17.

Lemma 19 *Consider a sequence of multigraphs $G(1) \subseteq G(2) \subseteq \cdots \subseteq G(k)$ with pairs $\{(s_i, t_i) : i \in [k]\}$. Assume that for each $i \in [k]$, the sparsity of multigraph $G(i)$ with demands $\{(s_j, t_j)\}_{j=1}^i$ is at least $\log i$. Then the sparsest priority cut is at least $\Omega\left(\frac{1}{\log\log k}\right)$.*

Proof Sketch. This is almost identical to the proof of Lemma 10. The only difference is in inequality (1) which becomes:

$$N(C) \quad \leq \quad \sum_{h=1}^{\ell} N(C_h) + \frac{1}{\log j} \cdot \sum_{h=2}^{\ell} Q_h. \tag{9}$$

Recall that here C is a connected component that forms at time $j \leq k$ due to the merging of components $\{C_h\}_{h=1}^{\ell}$ in graph $H(j-1)$. Also C_1 is the component in $\{C_h\}_{h=1}^{\ell}$ with maximum terminals. This change in the recurrence for $N(\cdot)$ in turn affects the total contribution of each edge $e \in Q$ in the expansion of $\sum_{D:comp(H(k))} N(D)$. The number of times e contributes remains $\beta \leq 2\log_2 k$. Let $2 \leq T_1 \leq T_2 \leq \cdots \leq T_\beta \leq 2k$ denote the number of terminals in e's component whenever e contributes. Recall that the number of terminals in e's component (at least) doubles whenever e contributes in the recursion, i.e. $T_{b+1} \geq 2 \cdot T_b$ for all $1 \leq b < \beta$. Also, note that if e's component has T_b terminals then the total number of terminals at that time $j_b \geq T_b$. So e's total contribution is:

$$\sum_{b=1}^{\beta} \frac{1}{\log T_b} \quad \leq \quad \sum_{b=1}^{\beta} \frac{1}{\log (2^b)} \quad = \quad \sum_{b=1}^{\beta} \frac{1}{b} \ \leq \ \log(2\log k).$$

This completes the proof of Lemma 19.

Lemma 20 *Consider the random priority multicommodity flow instance $\mathcal{M}$ seen in Step 3 of the online algorithm: sequence $G(1) \subseteq G(2) \subseteq \cdots G(k)$ of multigraphs with request-pairs $\{(s_i, t_i)\}_{i=1}^{k}$. The expected dynamic cost of routing on $\mathcal{M}$ is $O_\alpha(1) \cdot \log^{3\alpha+1} k \cdot (\log\log k)^{2\alpha} \cdot \mathsf{Opt}$.*

Proof: Consider any outcome w of the hallucination random process. We may add w to notation to make the outcome in consideration clear. As discussed to prove the lemma, it suffices to show a witness priority flow $\mathcal{F}^w$ that consumes dynamic power $O_\alpha(1) \cdot \log^{3\alpha+1} k \cdot (\log\log k)^{2\alpha} \cdot \mathsf{Opt}$ in expectation. Then the waterfilling algoirthm will find a priority flow that uses dynamic power within $O_\alpha(1)$ factor. Let $\mathsf{SP}(G_S(k))$ and $\mathsf{DP}(G_H^w(k))$ denote the static power of $G_S(k)$ and the dynamic power of $G_H^w(k)$, respectively. Note that $G_S(k)$ has no dependency on hallucination, hence is deterministic. Let I be the random variable denoting the maximal index $i \in [k]$ that *does not* satisfy the sparsity condition in Lemma 18. We will construct two separate priority flows on $\mathcal{M}$: flow $\mathcal{F}_1^w$ for pairs indexed more than I and $\mathcal{F}_2^w$ for pairs indexed up to I, and will set $\mathcal{F}^w := \mathcal{F}_1^w + \mathcal{F}_2^w$. We will show that $\mathcal{F}_1$ has dynamic power

$$O_\alpha(1) \cdot \log^{2\alpha} k \cdot (\log\log k)^{2\alpha} \Big(\log^\alpha k \cdot \mathsf{SP}(G_S(k)) + \mathsf{DP}(G_H^w(k)) \Big) \tag{10}$$

with probability one. We will also show that $\mathcal{F}_2^w$ has dynamic power

$$O_\alpha(1) \cdot \mathsf{SP}(G_S(k)) \tag{11}$$

in expectation. This would imply that that $\mathcal{F}_1 + \mathcal{F}_2$ has expected dynamic power at most $O_\alpha(1) \cdot \log^{2\alpha} k \cdot (\log\log k)^{2\alpha} \Big(\log^\alpha k \cdot \mathsf{SP}(G_S(k)) + \mathsf{DP}(G_H(k)) \Big)$; in this combination another 2^α factor loss can occur, but it is subsumed by $O_\alpha(1)$. Since the $\mathsf{SP}(G_S(k))$ is $O(\log k)\mathsf{Opt}$ and $\mathsf{DP}(G_H^w(k))$ can be easily shown to be $O(\log^\alpha k)\mathsf{Opt}$ from Lemma 1, this will prove the lemma.

Constructing flow $\mathcal{F}_1$. By definition of I, if we restrict attention to pairs $I+1, \ldots, k$ then the assumption in Lemma 19 holds. So the sparsest priority cut for the instance $G(I+1) \subseteq G(I+2) \subseteq \cdots \subseteq G(k)$ is $\Omega\left(\frac{1}{\log\log k}\right)$. By the priority flow-cut gap (Theorem 11) we obtain a priority multicommodity flow $\mathcal{F}_0^w$ of value $\frac{\Omega(1)}{(\log k \cdot \log\log k)^2}$ satisfying the virtual capacities. Let $\{h_e\}$ denote

the amount of hallucinated flow on each edge $e \in G_H^w(k)$. Recall that the virtual capacity $c_k(e)$ of any edge e equals h_e (if $e \in G_H^w(k)$) plus $\alpha \log k \cdot q$ (if $e \in G_S(k)$). So the dynamic power corresponding to the virtual capacities is:

$$\sum_{e \in G_F(k)} c_k(e)^\alpha \quad \leq \quad 2^\alpha \Big(\sum_{e \in G_H(k)} h_e^\alpha + \sum_{e \in G_S(k)} (\alpha \log k)^\alpha \cdot \sigma \Big) \quad \leq \quad O_\alpha(1) \Big(\mathsf{SP}(G_S(k)) + \mathsf{DP}(G_H^w(k)) \Big)$$

We define flow $\mathcal{F}_1^w$ to be $\mathcal{F}_0^w$ scaled by $O(\log k \cdot \log\log k)^2$ which is a valid priority flow for pairs indexed more than I. This proves the upper bound on the dynamic power of $\mathcal{F}_1^w$ claimed in (10).
Constructing flow $\mathcal{F}_2$. This simply uses the edges in the Steiner forest $G_S(k)$ to route all the pairs up to index I. The expected dynamic power on any edge $e \in G_S(k)$ is at most:

$$\mathbb{E}[I^\alpha] \quad = \quad \sum_{i=1}^{k} \Pr[I = i] \cdot i^\alpha \quad \leq \quad \sum_{i=1}^{k} \frac{1}{i^{2\alpha}} \cdot i^\alpha \quad = \quad O_\alpha(1).$$

The inequality uses Lemma 18. Thus the expected dynamic power of $\mathcal{F}_2^w$ is at most $O(1)$ times the static power of $G_S(k)$ as claimed in (11). This completes the proof. □

5 The Single-Source Case

We show that the online algorithm, with λ set to be $\Theta(\log k)$, is $O(\log^{\alpha+1} k)$-competitive for the single source case. Following the analysis in the previous sections, it is sufficient to prove that there is a capacity-respecting priority routing within the backbone $G_F(k)$.

Lemma 21 *With probability of at least $1 - k^{-2\alpha}$ there exists a capacity-respecting priority routing within $G_F(k)$.*

Proof: Let L_i be the set comprised of the source s and the sinks that have hallucinated by the time sink t_i arrived. In [6] (c.f. Lemma 1), it was shown that with probability of at least $1 - k^{-2\alpha}$, it is the case that for each sink t_i there exists a path P_i with the following properties:

- each path P_i goes from the sink t_i to a sink in $\ell_i \in L_i$ in the Steiner backbone $G_S(i)$, and then from ℓ_i to source s along the path taken by the hallucinated flow emanating from t_i,
- each edge of the Steiner backbone G_S is used by $O(q \log k)$ such paths, and
- each node in L_i is used by t $O(q)$ such paths.

This is a priority capacity-respecting routing. □

6 Capacitated Multicommodity Network Design

In this section, we consider the capacitated multicommodity design problem, as studied by [**?**]. In this problem, we are given a graph $G = (V, E)$ with each edge having a cost c_e and 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 multicommodity flow of the request-pairs.
Andrews et al. [**?**] consider this problem and use the techniques of embedding expanders via cut-matching games [**?**] and expander routing [**?**] to give a (polylog, polylog) bicriteria approximation algorithm for this problem. That is, the cost of their subgraph H is polylog times the optimal cost, and H can support 1/polylog flow of each request-pair concurrently. In this paper, we show the following theorem.

Theorem 22 *There is an efficient* $(O(1), O(\log k))$ *bi-criteria approximation algorithm for* CapND *when all edges have the same capacity.*

A related problem is that of survivable network design, where the requirement is H to satisfy the flow requirement, *individually for each demand* rather than concurrently.
We now present the details of our algorithm. Firstly, we argue that the case when the demands and capacities are the same is easy to approximate, by randomized rounding a natural LP relaxation (which allows violation in capacities by $O(\log k)$).

$$\min \sum_e c_e x_e \mathsf{LP_{cap}} \tag{12}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}_i} f(p) \geq 1 \qquad \forall i \in [k] \tag{13}$$

$$\sum_{p|e \in p} f(p) \leq x_e \qquad \forall e \in E \tag{14}$$

$$0 \leq f(p) \leq 1 \qquad \forall i \in [k], \forall p \in \mathcal{P}_i \tag{15}$$

$$0 \leq x_e \leq O(\log k) \qquad \forall e \in E \tag{16}$$

If the above LP is feasible, then we can do a simple randomized rounding (of its flow paths) for each request-pair, and get that the expected cost is at most the LP cost, and each edge is used to capacity at most $O(\log k)$.

Algorithm for CapND**:**

1. Constructing the Steiner backbone: Setting the cost of an edge to be c_e, we first buy an (approximate) minimum-cost Steiner forest [2,13]. We denote the set of edges bought in this step as the Steiner backbone G_S.

2. Constructing the Hallucination backbone: Each request-pair (s_i, t_i) independently hallucinates a demand of q, with probability $p = \min\{1, 32\lambda/q\}$. Here $\lambda = \Theta(\log k)$ is the flow-cut gap for multicommodity flow [21]. Now, using the procedure described above, solve the case of the problem where the hallucinated request-pairs have demand of q and edges have capacity q. If the LP was infeasible, redo this step. Let the subgraph bought be G_H, and let $\mathcal{H}$ denote the hallucinated routing.

3. Final Solution: Return the subgraph $G_S \cup G_H$.

6.1 Analysis

The analysis proceeds along the following lines, and is similar to that in Section 3.

Cost of G_S: By using the Steiner forest approximation algorithm from [2,13], one can guarantee that the cost of the edges in G_S is at most twice the minimum cost to even achieve connectivity between the request-pairs, and is hence at most $2 \cdot \mathsf{Opt}$.

Cost of G_H: We can first apply a standard concentration bound to argue that, w.h.p, the demand of the hallucinated request-pairs can be routed on the support of Opt while violating the capacities by a factor of $O(\log k)$, giving us a feasible LP solution to $\mathsf{LP_{cap}}$. Then our rounding

algorithm for the case when demands and capacities are equal returns a graph G_H which can unsplittably route the scaled-up demands of the hallucinated request-pairs (violating the capacities by $O(\log k)$). Recall that we call this unsplittable routing $\mathcal{H}$.

Routing on $G_S \cup G_H$: The last part of the proof shows there is a flow $\mathcal{F}$ of low congestion in the subgraph G_F. To this end, we assign *virtual capacities* $\{\hat{c}_e : e \in E\}$ to each edge in G_F equal to the amount of hallucinated flow routed on it in $\mathcal{H}$, plus $O(\lambda q)$ if it is in the Steiner backbone G_S. Using these virtual capacities, we can use Lemma 2 to conclude that w.h.p, the sparsity of every cut is at least λ (w.r.t the original demands of the 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 [20]. Finally, the virtual capacity of each edge is $O(\log k)q$ because $\mathcal{H}$ uses any edge to at most $O(\log k)q$, and we added $O(\log k)q$ units for edges in G_H.

If the demand flow is not routable with $O(\log k)$ violation in capacities, we repeat the algorithm. This completes the proof of Theorem 22.

7 Conclusion

Say our results extend to splittably routing variable demands.
Say something about extending our techniques to unsplittably routing request-pairs with varying demands using techniques in [14].
Ravi, Do you want to add something here about other routing problems that our techniques solve?

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

[6] 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.

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

[8] 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.

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

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

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

[12] 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.

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

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

[15] 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.

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

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

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

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

[20] 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.

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

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

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

[24] 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.

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

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

A Probabilistic Inequalities

Theorem 23 ([22]) *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 24 ([16,24]) *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 25 ([6]) *Let $p \geq 0$, and let $X_1, X_2, \ldots, X_n$ be i.i.d. random variables taking value D with probability $\max\{1, p\}$. 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 24 is $K_\alpha \max(pND, (pN)^{1/\alpha}D)$. Taking α^{th} powers and replacing the max by a sum, we get $\mathbb{E}[(\sum_i X_i)^\alpha] \leq (K_\alpha\, D)^\alpha((pN)^\alpha + pN)$. □