The All-or-Nothing Multicommodity Flow Problem

Chandra Chekuri
Lucent Bell Labs
600 Mountain Avenue
Murray Hill, NJ 07974.
chekuri@research.bell-labs.com

Sanjeev Khanna*
Dept. of Comp. & Inf. Sci.
University of Pennsylvania
Philadelphia, PA 19104
sanjeev@cis.upenn.edu

F. Bruce Shepherd
Lucent Bell Labs
600 Mountain Avenue
Murray Hill, NJ 07974.
bshep@research.bell-labs.com

ABSTRACT

We consider the all-or-nothing multicommodity flow problem in general graphs. We are given a capacitated undirected graph $G = (V, E, u)$ and set of k pairs $s_1t_1, s_2t_2, \ldots, s_kt_k$. Each pair has a unit demand. The objective is to find a largest subset S of $\{1, 2, \ldots, k\}$ such that for every i in S we can send a flow of one unit between s_i and t_i. Note that this differs from the *edge-disjoint path* problem (EDP) in that we do not insist on integral flows for the pairs. This problem is NP-hard, and APX-hard, even on trees. For trees, a 2-approximation is known for the cardinality case and a 4-approximation for the weighted case. In this paper we build on a recent result of Räcke on low congestion oblivious routing in undirected graphs to obtain a poly-logarithmic approximation for the all-or-nothing problem in *general* undirected graphs. The best previous known approximation for all-or-nothing flow problem was $O(\min(n^{2/3}, \sqrt{m}))$, the same as that for EDP. Our algorithm extends to the case where each pair s_it_i has a demand d_i associated with it and we need to completely route d_i to get credit for pair i. We also consider the *online admission control* version where pairs arrive online and the algorithm has to decide immediately on its arrival whether to accept it or not. We obtain a randomized algorithm with a competitive ratio that is similar to the approximation ratio for the offline algorithm.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.2.1 [**Discrete Mathematics**]: Combinatorics.

General Terms

Algorithms, Theory.

*Supported in part by an Alfred P. Sloan Research Fellowship and by an NSF Career Award CCR-0093117.

STOC'04, June 13–15, 2004, Chicago, Illinois, USA.

Keywords

All-or-Nothing Multicommodity Flow, Approximation Algorithms, Edge Disjoint Paths, Multicommodity Flow, Oblivious Routing, Online Algorithms.

1. INTRODUCTION

1.1 Background

A pervasive problem in communication networks is that of allocating bandwidth to satisfy a given collection of service requests. In situations where there is limited network capacity but an abundance of requests, one must optimize over the choice of which requests to satisfy. Such *maximization* problems arise for instance in the area of bandwidth trading, or when operators carve out subnets (so-called VPNs) within their network, for sale to interested enterprise customers.

Constraints on how bandwidth may be allocated vary according to the type of requesting service, as well as the technology in the underlying network. In SONET networks for instance, each request must reserve a path between its origin and destination, each link of the path supporting a traffic rate specified by the request. (In this paper, we ignore any restrictions imposed by requirements that traffic be protected against network failures.) Depending on the scale of demands, point-to-point traffic is often aggregated and designed for in bulk. In this setting, it makes sense to allocate bandwidth in terms of flows, rather than paths. Similarly, in many data networks, fractional routing is used to specify probabilities for routing traffic to a fixed destination.

In this paper, we discuss and compare several fundamental models related to this class of optimization problems. In each model we are given a n vertex capacitated graph $G = (V, E, u)$ (undirected or directed); here u denotes a non-negative integer edge (we use edge to also refer to directed arcs) capacity vector. In addition, we are supplied with a set of k (unit) demand node-pairs $s_1t_1, s_2t_2, \ldots, s_kt_k$, each possibly with its own weight w_i. We call the s_i's and t_i's *terminals* and note that they need not be distinct. The objective is to find a largest (or maximum w-weight) *routable* subset S of $\{1, 2, \ldots, k\}$, that is, the demands in S can be simultaneously satisfied obeying the capacity of the graph. We call such a set, *routable*. It is how routability is defined, that distinguishes the different models.

The most basic model is the *edge-disjoint path problem*, EDP. Here, a set S is routable if G contains edge-disjoint paths P_i, for each $i \in S$, such that P_i joins s_i and t_i. In the directed graph setting, we require that P_i is a directed path from s_i to t_i. For this problem the best approximation ratio available is $O(\min(n^{2/3}, \sqrt{m}))$ for undirected graphs [11] and $O(\min((n \log n)^{2/3}, \sqrt{m}))$ for directed graphs [36]. The directed version of this problem is provably hard to approximate. In [18], it was shown that for any fixed $\epsilon > 0$, there is no $O(n^{1/2-\epsilon})$-approximation algorithm unless P=NP. The story for undirected (EDP) is incomplete however. Even though the approximation ratio is polynomial in n, the best known inapproximability bound states that the problem is APX-hard [17]. Closing this gap is one of the fundamental open problems in approximation algorithms. Such a dichotomy in the hardness, or our understanding, between the directed and undirected versions is interesting and occurs frequently. Examples include connectivity problems and the sparsest cut problem. Partly in response to this dichotomy for EDP, we focus on a class of maximization problems where demands only request a (fractional) unit flow in the network. This forms a relaxation of EDP and since our positive bounds improve on the best known for EDP (we establish a poly-logarithmic approximation), we believe it gives impetus to resolving the situation for undirected EDP.

1.2 All-or-Nothing Multicommodity Flows

For the remainder of the paper we study the *all-or-nothing maximum multicommodity flow problem* which we denote by AN-MCF. For this version, a set S is routable if there is a multicommodity flow in G that satisfies every demand $i \in S$. In other words we want to find the largest weight subset of S such that the maximum concurrent multicommodity flow for the subset is at least 1. This differs from the *maximum edge-disjoint path* problem (EDP) in that we do not insist on integral flows for the pairs. We also observe that given S, the problem of deciding whether all of S can be routed, is poly-time solvable via linear programming while this same decision problem is NP-hard for EDP (even for undirected graphs and the demands lie amongst a set of four terminals [15]).

In trees, the AN-MCF problem coincides with the maximum integer multicommodity flow problem. This problem on trees is APX-hard [17] and a 2-approximation is known for the cardinality case (each $w_i = 1$) [17] and a 4-approximation for the weighted case [12]. For general graphs, where fractional routings come into play, there is no previous work on AN-MCF. The best known approximation factor for the general problem so far is $O(\min(n^{2/3}, \sqrt{m}))$, the same as that for EDP [11]. We also note that there are no simple or obvious algorithms that can take advantage of the fractional routing as the APX-hardness on trees demonstrates.

A natural linear programming relaxation (which we denote by MCF-LP) for the all-or-nothing problem is as follows. For each demand pair i, let $\mathcal{P}_i$ denote the paths joining s_i, t_i in G. We then have a nonnegative variable $x(P)$ for each $P \in \mathcal{P}_i$ and each $i = 1, 2, \ldots, k$. The objective is to maximize $\sum_i w_i \sum_{P \in \mathcal{P}_i} x(P)$ subject to the constraints $\sum_{P \in \mathcal{P}_i} x(P) \leq 1$ for each demand i, and $\sum_{P : e \in P} x(P) \leq u_e$ for each edge e. Note that this same LP is also valid for EDP and in fact all known approximation algorithms for EDP obtain their ratios directly or indirectly via the lower bound provided by the LP. Throughout, we let OPT denote the optimal value for this LP for some fixed instance. Our main result is the following.

THEOREM 1.1. *There is a poly-time algorithm which given an undirected n-node instance of* AN-MCF *problem, returns a solution with weight $\Omega(\frac{\text{OPT}}{\log^3 n \log\log n})$.*

In contrast to the above, an $\Omega(\sqrt{n})$ integrality gap is known for this LP applied to EDP [17]. Our proof of Theorem 1.1 builds upon the recent result of Räcke on decomposing undirected graphs [30] to obtain a *demand oblivious* routing that yields a poly-logarithmic approximation for the congestion version of multicommodity flow problems. Specifically, for a given n-vertex capacitated graph G, Räcke constructs a fractional 1-flow for every pair of nodes. He shows that such a *fixed routing* exists with the following striking property: given *any* demand matrix on the vertices, routing the flow according to the fixed routing (that is, if a pair st has a demand d, the d units of flow are routed along the fixed flow paths for the unit flow from s to t with each path getting d times its share of the unit flow), the congestion on any edge is $O(\log^3 n)$ times the optimal congestion for routing the given demand matrix. We use these *demand oblivious* routings of Räcke in the present setting, although we must compose them with other routings to avoid bottlenecks in lightly capacitated regions of the network. Azar et al. [6] and Applegate and Cohen [1] show that the *optimal* (with respect to congestion) oblivious routing can be computed in polynomial time by solving a linear program. Bansal et al. [7] give an online algorithm to compute a near optimal oblivious routing. However we need the tree based hierarchical scheme of Räcke in our algorithm. We note that some other applications of oblivious routing schemes also need the tree based hierarchical decomposition. See Maggs et al. [29] for an application to speed up iterative solvers of linear systems.

We have thus far assumed that each demand i asks for a "unit" flow between its endpoints, but on occasion we may be supplied more generally with a demand d_i other than 1. We mention that our techniques equally apply to such *multicommodity demand flow problems* (AN-DMCF). Here, a subset S of the demands is *routable* if there exists a multicommodity flow in G that satisfies each demand $i \in S$. Note that we maintain the all-or-nothing aspect that we receive the credit w_i for demand i only if we fulfill the whole demand d_i. For an instance of the demand flow problem, we let $d_{\max} = \max\{d_i : i \in S\}$ and $u_{\min} = \min\{u_e : e \in E\}$. When $d_{\max} \leq u_{\min}$ (no *bottleneck* case), our result for unit demand carries over with a loss of a constant factor in the approximation ratio by using a result in [12]. On the other hand, when $d_{\max}$ and $u_{\min}$ are allowed to be arbitrary, it was noted in [18], that the demand flow problem cannot in general be approximated better than $O(n^{1/2-\epsilon})$ if integer flows are required. However the proof extends directly to show the same hardness even if fractional flows are permitted. In [9], the integrality gap of the LP for demand flow is shown to be $\Omega(n)$ even when G is a path. But if we allow an additive congestion of $d_{\max}$, we can once again carry over our results, obtain the following extension of Theorem 1.1. Recall that OPT is the value of an optimal solution to MCF-LP to the given instance.

THEOREM 1.2. *There is a poly-time algorithm which given an undirected n-node instance of the multicommodity demand flow problem (*AN-DMCF*), returns a solution with weight at least $\Omega(\frac{\text{OPT}}{\log^3 n \log\log n})$ such that the flow on each edge e is at most $u(e)$ if $d_{\max} \leq u_{\min}$, and is bounded by $u(e) + d_{\max}$ when $d_{\max}$ and $u_{\min}$ are allowed to be arbitrary.*

Admission Control in the Online Setting: We have defined the AN-MCF and AN-DMCF problems as offline optimization problems. We also consider the *online* versions of these problems. We are given a capacitated graph G upfront, however the pairs that need to be routed arrive online. When a pair st is presented to the online algorithm, it has to immediately decide if it should accept or reject the pair, and if it does, has to route one unit of flow from s to t (d_i units in the demand case). For the unit demands case, the competitive ratio of the algorithm is the worst case ratio of the number of demands routed by the online algorithm to the optimal offline algorithm. For non-unit demands, we compare the sum total of demands routed (throughput) by the online algorithm to the flow routed by the optimal offline algorithm. We assume that once the pair is accepted it stays forever. In the routing literature this model is referred to as the permanent connection model [31]. We have the following theorem.

THEOREM 1.3. *For the* AN-MCF *and* AN-DMCF *problems there are randomized online algorithms that route* $\Omega(\frac{\text{OPT}}{\text{polylog}(n)})$ *flow such that the flow on each edge* e *is at most* $u(e)$ *if* $d_{\max} \leq u_{\min}$*, and is bounded by* $u(e) + d_{\max}$ *when* $d_{\max}$ *and* $u_{\min}$ *are allowed to be arbitrary.*

We mention that all our results improve by a $\log n$ factor for planar graphs or graphs that exclude minors of fixed size.

1.3 Related Work

The all-or-nothing flow problem, as far as the authors are aware, was essentially first raised in [12] in the context where the supply network is a tree. Of course, in the tree case, there is no advantage to allowing fractional routing: each commodity must route all of its demand on a single path. General multicommodity flow problems that require each demand to be routed on a single path, are called *unsplittable flow problems*, EDP is of course a special case of UFP where all demands are 1. Multicommodity flow, EDP, UFP, their generalizations, and their special cases have been extensively studied both for their fundamental importance to combinatorial optimization and their applications to a variety of areas such as network routing, VLSI layout, parallel computing and many others. It is infeasible to do justice to the literature due to space constraints, hence we confine ourselves to mentioning the directly relevant literature on EDP and UFP and their online variants. We refer the reader to [32, 25, 16, 21, 31, 35, 18, 23, 30, 11] for some pointers.

We first consider the offline case. For both EDP and UFP the best known approximation ratios in general graphs are $O(\min(n^{2/3}, \sqrt{m}))$ for undirected graphs [11, 24, 35] and $O(\min((n \log n)^{2/3}, \sqrt{m}))$ for directed graphs [36, 35]. EDP and UFP are APX-hard in undirected graphs [17] and are $\Omega(n^{1/2-\epsilon})$-hard in directed graphs [18]. If all pairs share a source, then EDP reduces to the one commodity maximum flow problem and can be solved in polynomial time. Even for UFP, the single source case is tractable in that most variants have constant factor approximation algorithms [20, 22, 14]. For EDP and UFP, large capacities help. Randomized rounding [32] yields constant factor approximation algorithms if $d_{\max} \leq u_{\min}/(\Omega(\log n))$. More generally if $d_{\max} \leq u_{\min}/B$ for integer B, then an approximation ratio of $O(Bn^{1/B})$ is achievable [35, 5]. We mention that all of the offline bounds for EDP and UFP also apply to the integrality gap of MCF-LP. It is known however that MCF-LP has an integrality gap of $\Omega(\sqrt{n})$ [17].

Now we consider the online versions of EDP and UFP. These problems have applications in ATM networks and are usually referred to as admission control for virtual circuit routing. A variety of models exist based on whether the circuits (pairs) are permanent or temporary and in the temporary case whether their durations are known or unknown. We refer the reader to the survey [31] for more details. Here we confine ourselves to the case of permanent connections. In this case online EDP asks to maximize the number of pairs accepted compared to the offline optimal. For UFP we are interested in maximizing the *throughput*, that is the sum of demands accepted. For both these problems the best competitive ratios are the same as the approximation ratios mentioned above in the offline setting. However, if capacities are large relative to the demands (a more realistic assumption in practice) the algorithm of Awerbuch, Azar, and Plotkin [4] is $O(\log n)$-competitive provided $d_{\max} \leq u_{\min}/\Omega(\log n)$.

In the context of EDP and UFP we can also consider the problem of routing a given set of demands S so as to minimize the *congestion* of the routing. Congestion of a routing is defined as the maximum over all edges of the ratio of the flow on the edge to its capacity. If flows can be split, then the minimum congestion routing can be computed by solving a linear program. However for integral flow paths or for unsplittable flow, randomized rounding is the only effective algorithm known and yields an additive approximation of $O(\log n/ \log\log n)d_{\max}$ on the optimal fractional congestion [32]. In directed graphs this gap is known to be tight [26]; that is there are instances where S can be fractionally routed with congestion 1 while any integral routing has $\Omega(\log n/\log\log n)$ congestion. Very recently Chuzhoy and Naor [13] have shown that, in directed graphs, the minimum congestion to route a given set of demands is hard to approximate within a factor of $\Omega(\log\log n)$. Despite this hardness, even in the case where demands to be routed arrive online, an $O(\log n)$-competitive ratio is possible [2]. More recently, Räcke obtained a randomized $O(\log^3 n)$-competitive algorithm for minimizing congestion that is *oblivious*. The ratio has been improved to $O(\log^2 n \log\log n)$ by Harrelson, Hildrum, and Rao [19]. This remarkable result of Räcke is the starting point for our work.

2. PRELIMINARIES

2.1 Notation and Terminology

We consider multicommodity flow problems in a given capacitated graph $G = (V, E, u)$ and we assume that each capacity $u(e)$ is an integer. We let $n = |V|$ and $m = |E|$. Throughout, for any graph G and proper node subset $S \subseteq V$, we denote by $\delta_G(S)$, or simply $\delta(S)$ if G is clear from the context, the set of edges of G with exactly one endpoint in S.

An instance of *multicommodity flow* consists of a graph $G = (V, E, u)$ and a collection of undirected demand pairs $s_i t_i$ for $i = 1, 2 \dots, k$ where each $s_i, t_i \in V$. In addition, we may have a weight w_i associated with each demand. Let $\mathcal{P}_i$ denote the paths joining s_i, t_i in G. For each path P in G, let $x^i(P)$ denote a nonnegative variable $x(P)$. An assignment x is called a *multicommodity flow* and is said to *satisfy* demand i, if $\sum_{P\in\mathcal{P}_i} x^i(P) = 1$. (If in addition, a demand has an associated value d_i, then the right hand side changes from 1 to d_i.) The *load* of x on the edge e is $l(e) = \sum_{i,P:e\in P} x^i(P)$ and the *congestion* on e is $l(e)/u(e)$. The flow x is *feasible* if the congestion on each each edge is at most 1. A set S of demands is *routable* if there is a feasible flow that satisfies each demand $i \in S$. We often say that a demand i is *routed* on path P, if $x^i(P) > 0$, or the flow obtained by *routing* along certain paths.

Consider two nonnegative vectors π, π' defined on the nodes of G such that $L = \sum_v \pi(v) = \sum_v \pi'(v)$. By *distributing* (or *routing*) $\pi(v)$ units of flow from v to π', we refer to a (single-source) flow f with the property that for each v', there is a flow of value $\pi(v)\pi'(v)$ between v to v'. By distributing flow from π to π', we refer to a multicommodity flow such that for each v we are sending $\pi(v)$ units of flow from v to π'.

The objective of the all-or-nothing multicommodity flow problem is to find a maximum weight routable set. The natural relaxation for this problem is to find a feasible flow that maximizes $\sum_i \sum_{P\in\mathcal{P}_i} w_i x^i(P)$. [1]

For a graph G, we call a capacitated tree $T = (V_t, E_t, u_t)$ with a specified *root* node r_t, a *Räcke tree* for G if the leaves of T are precisely the nodes of G. In particular, r_t is not one of the leaves. Note that each edge $e \in T$, determines, in a natural way, a partition $S \cup (V - S)$ of G's nodes. The capacity of e, $u_t(e)$, is defined as the total capacity on edges in the cut $\delta_G(S)$. We also let $h(T)$ denote the height of T. Each Räcke tree induces a canonical routing strategy for G, independent of the set of demands. These *demand-oblivious routings* are discussed in more detail later; we describe now some of their properties.

The demand-oblivious routing strategy consists of a unit flow for each pair of nodes s, t. Let F_{st} denote such a flow satisfying one unit of demand between s, t. Given a collection of demands, we may obtain a flow satisfying them as follows. For each i, and each $P \in \mathcal{P}_i$, route $F_{st}(P)$ amount of flow along P. The multicommodity flow obtained is said to be *routed according* to the demand-oblivious routes, or to the Räcke routing.

For each Räcke tree T, there exists a value $\alpha(G, T)$ that measures the quality of routing according to the demand-oblivious routes. [2] We make this precise now. Consider a set X of "demands" in G, i.e., X is a multiset of undirected edges uv with $u, v \in V$. The *congestion* of X on an edge e of T is the congestion on e resulting from routing each demand of X along its unique path in T. We say X is *routable on T* if the congestion is at most 1 on each edge of T.

THEOREM 2.1 (RÄCKE [30]). *Let T be a Räcke tree for G and X a set of demands. If X is routable on T, then Räcke routing these demands in G results in congestion at most $h(T)\alpha(G, T)$ on each edge of G. Moreover, there exists a Räcke tree T with $\alpha(G, T) = O(\log^3(n))$ and $h(T) = O(\log n)$.*

Räcke's original construction did not yield a poly-time algorithm to find such a T. Subsequently, two independent papers [8] and [19] obtained poly-time algorithms. In [19] it is shown how to construct T with $\alpha(G, T) = O(\log^2 n \log\log n)$ and $h(T) = O(\log n)$.

The running times of these algorithms are not polynomially bounded in the size of instances with arbitrary capacities. For our present purposes, we can side-step this issue as follows. First, given an instance of AN-MCF, we may find a basic solution in polynomial time. One can argue that all but at most m demands are satisfied, and moreover routed unsplittably on a single path. We may keep these latter demands and then reduce the vector of edge capacities so as to be minimal subject to supporting the flow for the remaining fractionally routed demands. Since there are at most m such demands, these reduced capacities are polynomially bounded in $|V|, |E|$. In the following, we let $\alpha(G)$ denote the minimum of $\alpha(G, T)$ over some computable class of Räcke trees T for G after it has possibly been reduced as such.

2.2 Räcke Routing in More Detail

In this section we describe the oblivious routing strategy of Räcke [30] based on hierarchical decomposition of the given graph. We need a refinement of Räcke's result, and hence we delve into the details of how the demand-oblivious routings are constructed. In doing so, we follow the approach and methods of [8] which simplified that of [30], in addition to giving a polynomial time construction of the decomposition in [30].

Let $T = (V_t, E_t, u_t)$ be some Räcke tree for $G = (V, E, u)$. We denote by T_v the subtree of T rooted at v and let L_v denote the leaves of the subtree T_v and G_v, the subgraph of G induced by L_v. Recall then, that $u_t(e)$ equals the capacity of the cut $\delta(L_v)$ in G. We mention that a key feature of Räcke's routing is that any demand st with $s, t \in L_v$ will be entirely routed within the subgraph G_v. We now describe the oblivious routing of Räcke in detail.

First we set up some notation. Consider a non-leaf node $v \in V_t$ with descendants $v_1, v_2, \ldots v_k$. We deal with two important sets of edges induced by such a node v. The first is the set of edges in the cut induced by the leaves L_v, that is, $\delta_G(L_v)$. Second, we say an edge is *separated by v* if either it lies in $\delta(L_v)$ or it has its endpoints in distinct subtrees T_{v_i}, T_{v_j}. Let $S(v)$ denote the set of such edges. A measure of these edges' capacity is critical in the following; we denote by $U(v)$ the sum $\sum_i u(\delta(L_{v_i}))$. Note that the capacity of each edge in $S(v)$ is accounted for twice except for those in $\delta(L_v)$ which are accounted for once. For a vertex $a \in V(G)$ such that $a \in L_v$, we denote by $u_v^{\text{cut}}(a)$ is the set of edges incident to a that lie the cut $\delta_G(L_v)$. In other words, the quantity $u(\delta_G(a) \cap \delta_G(L_v))$. We denote by $u_v^{\text{sep}}(a)$ the quantity $u(\delta_G(a) \cap S(v))$. We denote by $\pi_v^{\text{cut}}(a)$ and $\pi_v^{\text{sep}}(a)$ the quantities $u_v^{\text{cut}}(a)/u(\delta_G(L_v))$ and $u_v^{\text{sep}}(a)/U(v)$ respectively. Note that $\sum_{a\in L_v} \pi_v^{\text{cut}}(a) = \sum_{a\in L_v} \pi_v^{\text{sep}}(a) = 1$.

We describe a strategy for routing traffic from a node $s \in V$ to a node $t \in V$. Let $P = (s = v_0, v_1, \ldots, v_p, v_{p+1}, \ldots, v_l = t)$ be the unique path in T joining s and t, where v_p is the "high point" of the path (the least common ancestor of s and t in T). We construct a unit flow from s to t by concatenating a collection of (multicommodity) flow vectors.

Each edge and each internal node of P sponsors a transformation of the flow. An edge (y, v) (where y is a child of v) transforms a supply flow vector π_y^{cut} one nodes of L_y into a demand vector π_v^{sep} one nodes of L_v. This is called a *spreading* transformation. In other words, it is distributing

[1] We prefer this path-formulation for flows to the compact formulation for ease of presentation. In any case, either formulation admits a poly-time algorithm to solve this relaxation.

[2] We note that α can be computed by solving $|V_t - V| \le n$ multicommodity flow problems in G.

flow from π_y^{cut} to π_v^{sep}. If v is a node with some parent node x, then v transforms a supply flow vector π_v^{sep} on the nodes of L_v into demands π_v^{cut} on the nodes of L_v. This is called a *concentrating* transformation. Note that the fractions π_v^{cut} and π_v^{sep} are independent of the choice of s, t. These transformations are accomplished by associated multicommodity flows $f_{(y,v)}$ (called the *spreading* flow) and f_v (called the *concentrating* flow) defined on G_v. The flow $f_{(y,v)}$ satisfies the requirements for the following demand matrix: for $a \in L_y$ and $b \in L_v$, $d_{(y,v)}(a,b) = \pi_y^{\text{cut}}(a) \cdot \pi_v^{\text{sep}}(b)$. The flow f_v satisfies the requirements for the following demand matrix: for $a, b \in L_v$, $d_v(a,b) = \pi_v^{\text{sep}}(a) \cdot \pi_v^{\text{cut}}(b)$. Note that $f_{(y,v)}$ and f_v satisfy the transformation requirements. Finally, for each y, v as above, we may also define $f_{(v,y)}$ to be a flow obtained by reversing all of the flow paths in $f_{(y,v)}$.

We now return to constructing a unit flow from s to t. This flow, denoted by $F_{s,t}$, is obtained from the path P by merging the flows $f_{(v_0,v_1)}$, f_{v_1}, $f_{(v_1,v_2)}$, f_{v_2}, $\ldots$, $f_{v_{p-1}}$, $f_{(v_{p-1},v_p)}$, $f_{(v_p,v_{p+1})}$, $f_{v_{p+1}}$, $f_{(v_{p+1},v_{p+2})}$, $f_{v_{p+2}}$, $\ldots$, $f_{(v_{l-1},v_l)}$.

At this point, we have not mentioned anything about the actual flow paths for these flow vectors. A key aspect of Räcke's strategy is to aggregate these flows for many different demand pairs. Indeed, he shows that in order to do this efficiently (w.r.t congestion) it is enough to have for each non-leaf node v in T, an efficient routing for some canonical problem in G_v. We call this the *exchange flow problem for* v. For each non-leaf node $v \in T$, we consider a multicommodity flow g_v in the graph G_v for the following *exchange flow problem.* For each pair of nodes a, b in L_v, there is a demand of $D_v(a,b) = u_v^{\text{sep}}(a) \cdot u_v^{\text{sep}}(b)/U(v) = \pi_v^{\text{sep}}(a) \cdot \pi_v^{\text{sep}}(b) \cdot U(v)$ (we write $D(a,b)$ if v is clear from the context). We are also given a *throughput guarantee* $q_v \le 1$ indicating that that we may route q_v times each of these demands simultaneously in the graph G_v (in other words the maximum concurrent flow for the demand matrix D_v).

The heart of Räcke's technique is to show that we may obtain our flow vectors $f_v, f_{(y,v)}$ from a solution for these exchange flow problems (after scaling by q_v). One may show that if some set of demands can be routed in T with congestion 1, then routing along the flow vectors $F_{s,t}$ obtained through the exchange flows has congestion $O(\alpha(G,T))$ in G.

For our purposes we need the following lemma regarding routing via Räcke trees.

Lemma 2.2. *Let T be a Räcke tree for G and let X be a set of demands that is routable in T and such that for each $st \in X$, the path joining s, t in T contains r. For $v \in V$, let $X(v)$ be the number of demands in X incident on v. Then, in G we can distribute $X(v)$ units of flow from each $v \in V$ to the Räcke distribution at r (that is, to $X(v)\pi_r^{sep}$) with congestion $\alpha(G,T)$.*

Proof. Consider the Räcke routing of X in G. For any pair $st \in X$, r is the least common ancestor. It follows that the Räcke routing for st consists of distributing one unit of flow from s to its Räcke distribution π_r^{sep} at r and similarly for t. Note that π_r^{sep} is agnostic to the terminal from which flow originates. Since X is routable in T with congestion 1, it follows that the congestion in G for Räcke routing X will be at most $\alpha(G,T)$. ∎

3. ALGORITHM

In this section, we present an $O((\alpha(G)\log n))$-approximation algorithm for the all-or-nothing multicommodity flow problem. We first develop a scheme that allows us to route a large fraction of demands with low congestion. Specifically, we show that for any $\epsilon(n) > 0$, we can obtain a solution of weight $\Omega(\frac{\epsilon(n)\text{OPT}}{\log^3 n \log\log n})$ such that the flow on any edge e is at most $\epsilon(n)u(e) + 1$. We then design a more sophisticated routing scheme that allows us to obtain a solution of weight $\Omega(\frac{\text{OPT}}{\log^3 n \log\log n})$ without exceeding any edge capacities.

In the remainder, we call an edge e of T λ*-light* if $u_t(e) < \lambda$, otherwise it is λ*-heavy.* We extend this to edges in G – an edge $(x,y) \in E$ is λ*-light* if for some $v \in T$, $e \in G_v$ and some λ-light edge in T is incident to v. If there is no such v, then e is λ-heavy.

3.1 Starting Point

We start by describing a simple algorithm that is the starting point of our approach:

(a) Construct a Räcke tree T for G.

(b) Scale down all tree capacities by setting $u'_t(e) = \max\{1, \lfloor u_t(e)/\alpha(G)\rfloor\}$. Let T' denote this new tree.

(c) Solve the all-or-nothing LP induced on T', and let $\text{OPT}(T')$ denote the value of an optimal LP solution.

(d) For the unweighted (weighted) problem, find a set X of size (weight) at least $\text{OPT}(T')/2$ [17] ($\text{OPT}(T')/4$ [12]) that can be routed integrally.

(e) For each routed pair $st \in X$, send 1 unit of flow from s to t in G using the flow function $F_{s,t}$ specified by the Räcke routing.

Lemma 3.1. *The algorithm above is a $2\alpha(G)$-approximation ($4\alpha(G)$-approximation) algorithm for the unweighted (weighted) all-or-nothing flow problem on G such that congestion is at most 1 on every $\alpha(G)$-heavy edge of G, and at most $\alpha(G)$ on every $\alpha(G)$-light edge in G.*

Proof. Let $\text{OPT}(G)$ denote optimal LP solution value for our instance of AN-MCF. It is clear that $|X|$ (weight of X) is at least $\text{OPT}(G)/2\alpha(G)$ ($\text{OPT}(G)/4\alpha(G)$ in the weighted case). Note that if X is feasible on T', then in T we can route α units of demand for each $i \in X$ with congestion at most 1 on the heavy edges, and at most α on the light edges. Thus in G we can route α demand for each pair in X with congestion at most α on α-heavy edges, and in fact Räcke's arguments imply congestion at most α^2 on α-light edges. Hence we can scale down and route a unit amount for each i in X with congestion 1 and α on α-heavy and α-light edges respectively. ∎

Corollary 3.2. *If all edges in T are $\alpha(G)$-heavy, there is a simple $O(\alpha)$ approximation algorithm for any instance of* AN-MCF *problem in G.*

3.2 Preprocessing the Instance

In the following we assume that X is a feasible set of demands for the tree T. For a node v of T, we denote by ℓ_v, its level (distance from the root) in the tree. For a demand $st \in X$ we denote by $\text{lca}(s,t)$, the level of the least common ancestor of s and t in T. We partition demands st according to their levels. Clearly we may find a subset $X' \subseteq X$ such that $|X'| \geq |X|/h(T)$ (or in the weighted case, $w(X') \geq w(X)/h(T)$) and all demands in X' have the same level. We focus on the set X' from here on. Let ℓ be the common level for pairs in X', i.e., ℓ is the level of T where all demands in X' "turn".

We describe next a procedure to decompose T and X into a disjoint collection of subtrees and pairs. Let $r_1, r_2, \ldots, r_p$ be the nodes of T at level l. For each i, let T_i denote the tree T_{r_i} and X'_i denote the demands of X' with both ends in T_{r_i}. We also let G_i be the subgraph of G induced by the leaves of T_i. Note that the graphs G_i are edge and vertex disjoint. Further, the LP on T for X routes X'_i entirely within G_i. For each i, in finding a subset of X'_i to route, we restrict ourselves to routing within G_i. Hence we can treat each i separately. We also note that T_i is a valid Räcke tree for G_i.

LEMMA 3.3. *Given an instance of* AN-MCF *on G with a Räcke tree T, with an $O(h(T))$ loss in approximation, we can restrict ourselves to instances in which for all demands $st \in X$, $lca(s,t) = r$.*

An instance (G', T', X') of AN-MCF is called NICE, if the X' can be routed on T with congestion 1 and each demand in X goes through the root. From our discussion above we obtain the following.

LEMMA 3.4. *Given an instance of (G,T,X) of* AN-MCF, *let* OPT *be the LP value of X on G. Then we can obtain $p \leq n$* NICE AN-MCF *instances (G_i, T_i, X_i), $1 \leq i \leq p$ such that $\sum_i |X_i| = \Omega(\frac{\text{OPT}}{h(T)})$.*

3.3 Low Congestion Routings for NICE Instances

Given a NICE instance (G,T,X) our goal is to find a large subset $Z \subseteq X$ that we can route in G with low congestion – a flow of $\epsilon(n)u(e)+1$ on each edge e for any given $\epsilon(n) > 0$. We start by establishing a simple lemma about grouping subsets of vertices in an edge-disjoint manner.

LEMMA 3.5. *Let G be a connected graph with a weight function $\rho : V \to [0, W]$ such that $\sum_{v \in V} \rho(v) \geq W$. We can find $p \geq \sum_{v \in V} \rho(v)/3W$* edge-disjoint *connected subgraphs $H_1 = (V_1, E_1)$, $H_2 = (V_2, E_2)$, ..., $H_p = (V_p, E_p)$ such that there exist* vertex-disjoint *subsets $S_1, S_2, ..., S_p$ such that for each i: (i) $S_i \subseteq V_i$ and (ii) $\sum_{v \in S_i} \rho(v) \geq W$.*

PROOF. Let $m := \sum_v \rho(v)$. The *weight* of a node subset X is $\sum_{v \in X} \rho(v)$ and X is called *heavy* if its weight is at least W. Let T be a spanning tree of G rooted at some node r. For any node x let T_x be the subtree rooted at x. Choose some x such that $V(T_x)$ is heavy, but none of x's children has this property (x exists since $m \geq W$). We group children of x together in a greedy fashion from left to right such that each group (which identifies some S_i's) has weight between W and $2W-1$. We may assume that x is put in the first of these groups. that the corresponding H_i's are formed from subtrees of children of x together with edges incident to x connecting these subtrees. Let j be the number of groups formed at x; by assumption $j \geq 1$. Thus we use up a total weight of at most $(2W-1)j$. The total weight of nodes, in T_x, not included in one of these H_i's is at most $W-1$. After grouping at x, we delete the edges in the subtrees corresponding to the H_i's. We also set $\rho(x) = 0$ and recurse on the remaining tree. At the end of the process we have a subset of weight up to $W-1$ that is never placed into any group. Hence if the total number of groups formed is ℓ, we have the inequality $\ell(2W-1)+W-1 \geq m$. We thus have that $\ell \geq \max(1, m/3W-1)$, and the result follows. ∎

Let S be the set of terminals in X. For a vertex $s \in V(G)$ let $X(s)$ be the number of demands in X incident on s.

LEMMA 3.6. *Let (G,T,X) be a nice instance. We can distribute $\frac{\epsilon}{\alpha(G)}X(s)$ flow from each $s \in S$ according to the Räcke distribution π_r^{sep} on $V(G)$ such that the flow on any edge $e \in G$ is at most $\epsilon u(e)$.*

PROOF. Following the proof of Lemma 2.2, we know that X can be routed in G, using Räcke routing, with a congestion of $\alpha(G)$. Scaling down the flow by $\alpha(G)/\epsilon$ and noting that each demand is distributing its flow to according to π_r^{sep}, gives us the desired result. ∎

We now identify a subset of $Z \subseteq X$ that can be routed in G with low congestion. With each $v \in V(G)$ we associate an integer counter $b(v) = \lfloor \frac{\epsilon}{\alpha(G)} X(v) \rfloor$ and and a weight $\beta(v) = \frac{\epsilon}{\alpha(G)} X(v) - b(v)$. Note that $0 \leq \beta(v) < 1$. Let $\beta = \sum_v \beta(v)$. We use Lemma 3.5 with $W = 1$ to group the terminals S into edge disjoint *clusters* $S_1, S_2, \ldots, S_p$ where $p = \max(1, \Omega(\beta))$ such that $\sum_{v \in S_i} \beta(v) \geq 1$. We reduce any $\beta(v) > 0$ until either it lies in a cluster S_i, $\sum_{v \in S_i} \beta(v) = 1$, or $\beta(v)$ becomes 0.

We greedily consider the pairs $st \in X$ one by one and build a set of demands Z which we ultimately route. Initially $Z = \emptyset$. We also maintain a set of *active* clusters: initially all clusters are active. Let st be the current pair. We say that the pair st is feasible at s if either $b(s) > 0$ or s is in an active cluster. We add st to Z if both s and t are feasible. Otherwise we reject the pair st. If st is added to Z we update as follows. If $b(s) > 0$ we decrement $b(s)$ by 1, otherwise we mark the cluster containing s as inactive. Similarly with t.

LEMMA 3.7. *This procedure produces Z with at least* $\max(1, \Omega(\frac{\epsilon}{\alpha(G)}|X|))$ *demands. Each cluster S_i, contains 0, 1 or 2 terminals for demands in Z. If it contains 2, then they are the end points of the same demand.*

LEMMA 3.8. *The subset Z can be routed in G such that on each edge e, the flow is at most $1 + \epsilon u(e)$.*

PROOF. Consider a pair st in Z. Suppose first that s, t lie in a common cluster. In this case we simply connect s and t by a path in this cluster. So we suppose this is not the case, and hence each of s, t will send a unit of flow to the root's distribution π_r^{sep}. If $b(s) > 0$ when st was added to Z, we route one unit from s to π_r^{sep}. Z, there was an active cluster S_i containing s, but not t. st was added to Z.

We then distribute one unit of flow from s to the vertices in S_i such that each vertex $v \in S_i$ gets a flow $\beta(v)$. This is feasible since $\sum_{v \in S_i} \beta_v = 1$. Each vertex $v \in S_i$ then sends $\beta(v)$ flow to the root's distribution π_r^{sep}.

We bound the congestion of an edge e as follows. The edge e can belong to at most one cluster. Within a cluster it can be used to route at most one unit of flow from a terminal. Now consider flow on e from the routings from terminals to the root. For v, let $f(v)$ be the total flow that is routed from v to π_r^{sep}. From our routing above it is easy to see that $f(v) \le b(v) + \beta(v) \le \frac{\epsilon}{\alpha(G)} X(v)$. Now we can apply Lemma 3.6 to claim that the flow on e is at most $\epsilon u(e)$. Hence the total flow on e is at most $1 + \epsilon u(e)$. ■

THEOREM 3.9. *Given an instance of* AN-MCF (G, X) *and a Räcke tree* T *there is a polynomial time algorithm that routes* $\Omega(\frac{\epsilon \text{OPT}}{h(T)\alpha(G,T)})$ *demands from* X *where* OPT *is the optimum LP solution for* X *on* G *such that the flow on any edge* e *of* G *is at most* $1 + \epsilon u(e)$.

Using [19] there is a Räcke tree T for G such that $\alpha(G,T) = O(\log^2 n \log\log n)$ and $h(T) = O(\log n)$; hence we obtain an approximation ratio of $O(\log^3 n \log\log n/\epsilon)$. For planar graphs or graphs that exclude a constant size minor, again using [19], we obtain an approximation ratio of $O(\log^2 n \log\log n)$.

3.4 No Congestion Routings for NICE Instances

We now show how to find a routing that does not violate the capacities. Again we work with NICE instances. Let (G, T, X) be a NICE instance. Let $X(v)$ be the pairs in X that are incident to a vertex $v \in G$. We say that a capacitated graph G is 2-edge connected if the uncapacitated multi-graph obtained by making $u(e)$ copies of each edge e is 2-edge connected. The main result of this section is the following.

THEOREM 3.10. *Given a* NICE *instance* (G, T, X) *of* AN-MCF *such that* G *is* 2*-edge connected, there is a polynomial-time algorithm that routes* $\max\{1, \Omega(|X|/\alpha(G,T))\}$ *pairs from* X *in* G *without violating capacities.*

The above theorem can be extended to arbitrary graphs by considering the 2-edge connected components of G to obtain the following.

THEOREM 3.11. *Given a* NICE *instance* (G, T, X) *of* AN-MCF*, there is a polynomial-time algorithm that routes* $\max\{1, \Omega(|X|/\alpha(G,T))\}$ *pairs from* X *in* G *without violating capacities.*

The rest of this section is devoted to the proof of Theorem 3.10. The basic idea is similar to that in Section 3.3. Given a nice instance (G, T, X) each of the X pairs can route $\epsilon/\alpha(G)$ flow such that each edge e has a flow of at most $\epsilon u(e)$. In Section 3.3 we used tree based clustering such that a terminal that we chose to route sends one unit of flow to $\Omega(\alpha(G)/\epsilon)$ other terminals. This additional routing required one unit of capacity on the edges of G. In this section we show that if G is 2-edge connected then we can cluster the terminals in such a way that a chosen terminal can send one unit of flow to $\Omega(\alpha(G))$ other terminals using at most 1/2 unit of capacity on the edges. Chosing $\epsilon = 1/2$ will ensure that no edge capacity is violated.

We are given an undirected 2-edge-connected graph G. There is also a nonnegative integer node vector ρ. We call $\rho(v)$ the *weight* of node v and call v a *terminal* in (G, ρ) if $\rho(v) > 0$. The ρ values are obtained by scaling up the flow at v such that all values are integers.

For a graph H and integer weight vector ρ', a single-source flow vector f from node v is *k-feasible* if the flow on any edge is at most $\frac{k}{2}$ and the total flow terminating at a node $y \neq v$ is at most $\rho'(y)$. A node $v \in V(H)$ is *central* for (H, ρ') if it is a terminal and there is a feasible flow from v of size $k - \rho'(v)$. We then call f, v a *k-center* of (H, ρ'). A node y is a *transmitter* in such a center if it is the source of the flow and $\rho'(v) < k$. It is called a *receiver* if y has positive net in-flow under f. We say that (H, ρ') is a *k-cluster* if $\sum_{v \in H} \rho'(v) \le 19k$ and it has a k-center.

We state below two simple lemmas that we use.

LEMMA 3.12. *Let* G *be a* 2*-edge-connected graph and* ρ *a nonnegative integer node-weight vector. For any* $v \in V$*, integer* k *and* $s := \min\{k, \sum_{u \neq v} \rho(v)\}$ *there is a* k*-feasible flow vector of value* s *with source* v.

LEMMA 3.13. *Let* G *be* 2*-edge-connected, and let* S_1, S_2 *induce* 2*-edge-cuts such that* $S_1 - S_2, S_2 - S_1$ *and* $S_1 \cap S_2$ *are nonempty. Then either* $S_1 \cap S_2$*, or both* $S_1 - S_2$ *and* $S_2 - S_1$ *induce* 2*-edge-cuts.*

Another useful lemma is the following.

LEMMA 3.14. *Let* G *be a* 2*-edge-connected graph,* ρ *a nonnegative integer node-weight vector and* v_1, v_2 *be a pair of distinct nodes. Let* $s_1 + s_2 = \sum_x \rho(x)$*. Then* G *contains an edge-disjoint pair of weighted trees* (T_i, ρ_i) *such that* v_i *lies in* T_i*, for each* v*,* $\rho_1(v) + \rho_2(v) = \rho(v)$*, and* $\sum_{v \in T_i} \rho_i(v) = s_i$ $(i \in \{1, 2\})$*. Moreover, we may assume* $\rho_2(v_1) = \rho_1(v_2) = 0$.

PROOF. We grow a tree T_1, ρ_1 rooted at v_1 as follows. At all times we maintain the property that the graph $G^{T_1} := G - E(T_1)$ has a component that includes v_2 and every node outside T_1. If the total weight of T_1 is less than s_1, we search for a new "good" edge to add to T_1 that maintains this property. If the total weight of nodes in $T_1 - v_2$ is greater than s_1, let z be the last node added to T_1. We obtain the desired tree by setting $\rho_1(z) = s - \sum_{v \in T_1} \rho(v)$. We start with the trivial tree rooted at v_1.

To complete the proof, it is enough to show that there is some good edge in $\delta(T_1)$. If an edge $e \in \delta(T_1)$ is not good, then it is a cut-edge in G^{T_1} that separates some node of $V - T_1$ from v_2. Consider the metric where each such edge has weight 1 and all others have weight 0. Choose an edge $e \in \delta(T_1)$ that is at a maximal distance from v_2 in G^{T_1} under this metric. In particular note that this implies that $G^{T_1} - e$ has two components one of which, say H, does not contain v_2 and but does contain a node $x \in V - T_1$. Since G is 2-edge-connected, there is a path from x to v_1 that does not contain e. Let f be the first edge on this path that crosses $\delta(T_1)$. Now if f is contained in H, then it must be good by maximality of e. Otherwise, the path must cross an edge f' in the cut $\delta(V(H))$ before entering $V(T_1)$. By assumption, $f' \neq e$, but any other edge of this cut lies in T_1, and so both ends of f' already lie in T_1, a contradiction. This completes the proof. ■

Our aim is to find a large collection of k-clusters in a general graph. From here on, we assume w.l.o.g. that $\rho(v) < k$ for each vertex v. Otherwise, we can repeatedly remove trivial k-clusters containing only the vertex v until $\rho(v)$ becomes less than k. We start with a simple lemma concerning clustering on certain graphs. We call an edge of a graph *short* if neither of its endpoints has degree exactly equal to 2.

LEMMA 3.15. *Let G, ρ be such that G has maximum degree 3 and its short edges form a matching amongst the degree 3 nodes, and each degree two node has positive weight. Then if $\sum_v \rho(v) \geq k$, we may partition into node-disjoint subtrees such that each subtree is a k-cluster, where nodes in a subtree receive their original weights.*

We now consider clustering in a general 2-edge-connected graph.

THEOREM 3.16. *Let G be a 2-edge-connected graph, k an integer and ρ a nonnegative integer node-weight vector such that $\sum_v \rho(v) \geq k$. Then we may find a collection of edge-disjoint k-clusters $(H_1, \rho_1), (H_2, \rho_2), \ldots, (H_p, \rho_p)$ such that for each node v, $\sum_i \rho_i(v) = \rho(v)$. Moreover, we may choose k-centers for these clusters so that each node either never appears as a transmitter, or never appears as a receiver.*

PROOF. Recall that we assume that $\rho(v) < k$ for each node v. Suppose that the statement is false and that G is a counterexample with a minimum number of edges. Suppose now that v is some node with degree at least 4. Then it is well-known [28] that there is a pair of edges e_1, e_2 incident to v that can be *split off* while maintaining 2-edge-connectivity. That is, if we let $e_i = u_i v$ for $i = 1, 2$, then the graph $G' = (G - \{e_1, e_2\}) \cup \{u_1 u_2\}$ is again 2-edge-connected. By minimality, G' contains the desired clusters. But if any cluster (H_i, ρ_i) used the new edge $u_1 u_2$, then clearly it can be extended to a k-cluster using the two edges e_1, e_2, and setting $\rho_i(v) = 0$ if v is not already in H_i.

Thus we may assume that every node in G is of degree 2 or 3. In addition, every degree 2 node is a terminal, or else we could suppress it to get a smaller counterexample. Now if every node is of degree 2, then G is a cycle and a desired collection of clusters can be found greedily. So we may assume that G is subcubic, that is, there is a cubic graph F such that G is obtained by subdividing some of the edges of F. For each $e \in E(F)$ let P_e be the path corresponding to this edge in G. We call an edge *short* if $|E(P_e)| = 1$. For a subgraph of F, its *image* (in G) is the subgraph obtained by replacing each e in the subtraph by P_e.

Note that if e is short and $F-e$ is 2-connected, then $G-e$ is also 2-connected and hence forms a smaller counterexample. So we assume no such edge exists and so every short edge lies in a 2-cut of F; call such an edge *bad*. We call a set $S \subseteq V(F)$ an *m-set* if it is a minimal set with the property that $|\delta_F(S)| = 2$. Since F is cubic, we have that $|S| \geq 2$. If there are no bad edges, then by Lemma 3.15, G could not be a counterexample. Thus there are some bad edges and hence some m-sets as well.

Lemma 3.13 implies that the m-sets are disjoint. Moreover, the uncrossing technique shows that for any m-set S, $F[S]$ contains no bad edges. For if $uv \in E(F)$ was bad and $u, v \in S$, then by assumption there is a set S' inducing a 2-cut containing uv. Uncrossing shows that either $S \cap S'$ or $S - S'$ again induces a 2-cut, contradicting minimality of S. This also implies that the collection of m-sets do not partition $V(F)$. If they did, then every bad edge has its endpoints in two distinct m-sets. But then since F is cubic and 2-edge-connected, the bad edges of F (and hence of G) form a matching, and so Lemma 3.15 contradicts G being a counterexample. In particular, this shows that if S is an m-set, then $|V - S| \geq 2$.

We now consider the global structure of m-sets. Consider the graph F' obtained by shrinking each m-set to a single node. Let the resulting set of shrunken nodes be called Y and for $y \in Y$, denote by S_y its corresponding m-set. From above, we have $X := V(F') - Y$ is nonempty. Since each $y \in Y$ is of degree 2, F' is obtained from the induced subgraph $F'[X]$ by adding internally node disjoint X-paths, i.e., paths starting and ending in X and each internal node in Y. Let $Q = v_0 v_1, \ldots, v_l$ be one of these paths and let G^Q be the subgraph that is the image of $F[S_{v_1} \cup \ldots \cup S_{v_{l-1}}]$.

We claim that we can find a 2-path Q so that deleting G^Q does not destroy 2-connectivity. To show the existence of a *removable* path consider the graph F'' obtained by replacing each 2-path by a single edge. F'' is now a cubic 2-connected minor of F (and hence G). Moreover, a 2-path P is not removable if and only if there is a 2-cut inducing set S in F'' that separates the endpoints of P. In particular, if there are no removable paths, then F'' contains 2-cuts, and hence contains some m-set. Let S be one such. We claim there is some 2-path Q whose endpoints y_1, y_2 both lie in S. If this were not the case, then we could find a 2-cut S' in F which is contained entirely in X, contradicting that all m-sets of F are contained in $V - X$. Consider the edge e' in F'' joining y_1, y_2. We claim that e' cannot lie in a 2-cut. If it did, then by uncrossing in F'' we would obtain a contradiction on minimality of S. Thus the 2-path Q is removable as required.

Let w_1, w_2 be the total weight of nodes in G^Q and $G - V(G^Q)$ respectively. If $w_1, w_2 \geq k$, then by minimality we may obtain clusters for G by clustering on G^Q and $G - V(G^Q)$ individually. Suppose first that $w_2 < k$. Consider the graph obtained by deleting all nodes outside of G^Q, except for v_0, v_l. We also assign a weight $w_2/2$ to each of v_0, v_l. By Lemma 3.15 we may cluster in this graph. Since v_0, v_l are leaves and have weight less than $k/2$, they may not be a center of any cluster. Thus by Lemma 3.14, we may appropriately extend any clusters by hanging trees from v_0, v_l.

Thus we may assume that $w_2 \geq k$ and $w_1 < k$. Consider the graph G' obtained from G by shrinking G^Q down to a single node v^*, which receives weight w_1. By minimality of G, we may find the desired clusters in G'. We consider several cases depending on how clusters send flow in or out of v^*. In the following, let s_1 be the total amount of flow that clusters place on edge $v_0 v^*$ and s_2 be the total flow $v^* v_l$. Note that each $s_i \leq k/2$. Case (I) *Clusters send flows in both directions to v^**. By assumption v^* can not be both a receiver and transmitter and so the only possibility is that v^* lies in a single cluster, but receives flow from, say, v_0 and some of this flow is sent further along the edge $v^* v_l$. Since $s_2 \leq s_1 \leq k/2$ we can cluster in G by greedily building trees on the images of $S_{v_1}, S_{v_2} \ldots$ and "dropping" off flow as we go along. Case (II) *Clusters only send flow into v^**. There may be $0, 1$ or 2 clusters sending flow into v^*. Let q be the smallest integer such that the weight of nodes in the image of $S_{v_1} \cup S_{v_2} \ldots \cup S_{v_q}$ is some $w \geq s_1$. Let

l, r be the nodes S_{v_q} that are incident to the two edges of $\delta_F(S_{v_q})$; without loss of generality $l \in S_{v_{q-1}}$ and $r \in S_{v_{q+1}}$. We may extend such clusters for G using Lemma 3.14 by greedily building trees hanging from v_0 and v_l as follows. We grow a tree T rooted at v_0 that spans each S_{v_j} with $j < q$. Next we apply Lemma 3.14 with $v_1 := l, v_2 := r$ and $s_2 = w - s_1$. It follows that adding T_1 to T results in a tree rooted at v_0 of weight exactly s_1. Similarly, we may build a tree rooted at v_l. Since each $s_i \le k/2$, this results in a legal clustering for G. Case (III) *Clusters only send flow from* v^*. In this case, v^* is central to some cluster. Let q be the smallest integer such that the weight of nodes in the image of $S_{v_1} \cup S_{v_2} \ldots \cup S_{v_q}$ plus s_1 is at least $k/2$. Let l, r be the nodes S_{v_q} that are incident to edges of $\delta_F(S_{v_q})$ as in case (II). Pick some terminal node y in the image of S_{v_q} to be our central node in a cluster for G. We apply Lemma 3.12 to the image of S_{v_q} in G using y as the root node. In addition, we increase the weight of l by s_1 plus the total weight of nodes in the images of S_{v_j}, $j < q$. Note that this increase is at most $k/2$ by choice of q. Similarly, we increase r by s_2 plus the weight of nodes in images of S_{v_j}, $j > q$. This increase is also at most $k/2$, since we may assume that $\rho(v^*) + s_1 + s_2 = k$. We find the desired flow from y to nodes in this image. We may now extend excess flow at l, r greedily. For instance, the excess flow into l is sent to nodes in the images of $S_{v_{q-1}}, S_{v_{q-2}}$ etc., and flow into r is sent in the other directions. ■

3.5 Multicommodity Demand Flows

When pairs in X have a demand associated with them, we can directly translate our result for the unit demand case to AN-DMCF using results in [12]. This gives us Theorem 1.2.

4. ONLINE ALGORITHM

We now describe an online version of our algorithm. We focus here on showing the online analog of the unit demand result in Section 3.3. We defer the extension to the no congestion case as well as the handling of the arbitrary demands to the full version of the paper.

Let T be a Räcke tree for a graph G, and let r be the root of T. For each leaf vertex x in T, let $P(x)$ denote the path from x to the root. We define *bottleneck ratio* for x, denoted $\beta(x)$, to be $\theta = \min_{(v,p(v)) \in P(x)} \frac{u_T(v,p(v))}{|L_v|}$ if $\theta < \alpha(G)/\epsilon$ and 0 otherwise. A vertex x with $\beta(x) > 0$ is refered to as a *bottleneck vertex*.

LEMMA 4.1. *Let T be a Räcke tree for a graph G. Then each vertex v can simultaneously distribute $\epsilon\beta(v)/\alpha(G)$ units of flow to $V(G)$ in accordance with the Räcke distribution at the root, such that congestion on any edge of G is at most ϵ.*

LEMMA 4.2. *Let T be a Räcke tree for a graph G, and let X be any collection of demand pairs such that each pair includes a bottleneck vertex in it and goes through the root. Then no algorithm can route more than $B = \sum_v \beta(v)$ of the pairs in G with congestion 1. Moreover, there is an online algorithm that can route a subset $Z \subseteq X$ of size at least $(\epsilon B)/(2\alpha(G))$ with congestion $1 + \epsilon u(e)$ in G.*

In the setup for the online algorithm, we construct a Räcke tree T for the input graph G. We then guess uniformly at random a level $\ell \in [1..h(T)]$. The online algorithm will only consider routing pairs st of requests such that $lca(s,t)$ is a level ℓ node in the tree. Thus the Räcke tree is implicitly partitioned into disjoint trees $T_1, T_2, \ldots$. In particular, note that we will then only consider requests with both endnodes in one of these trees.

Without loss of generality, from here on, we focus our attention on a single tree $T = T_i$ with root node r. For this tree, we create edge-disjoint connected subgraphs, using weight function β, as described in Lemma 3.5. If $x \in S_i$, we say that H_i is the cluster of x.

The online algorithm now proceeds as follows.

(a) Consider an arriving request st. Reject it outright if $\mathrm{lca}(s,t)$ is not r.

(b) We accept the request if each of s and t has either a path in T of residual capacity $\alpha(G)/\epsilon$ to the root, or an unused cluster.

(c) If the pair st is accepted, we proceed as follows. If s and t are in the same cluster we connect them via a path in the cluster. If s has a path in T of capacity $\alpha(G)/\epsilon$ to the root, we simply route in G, one unit from s to the root distribution π_r^{sep} in accordance with the Räcke routing. We then remove a capacity of $\alpha(G)/\epsilon$ along the relevant path in T. Otherwise, s distributes 1 unit of flow to vertices in its cluster in accordance with the weight function β. Each of the vertices in the cluster in turn send their flow to the root according to the Räcke distribution at the root. We do a similar routing for t. If we route using a cluster, we mark it as used.

Let Z be the set of demands routed by the above algorithm. Using similar arguments to the offline setting, one may show that $E[Z]$ is $\Omega((\epsilon\text{OPT})/(\alpha(G)\log n))$, and that the resulting congestion is $1 + O(\epsilon)u(e)$ on any edge.

5. REFERENCES

[1] D. Applegate and E. Cohen. Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs. *Proc. of SIGCOMM*, 2003.
[2] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *JACM*, 44(3):486-504, 1997. Preliminary version in *Proc. of STOC*, 1993.
[3] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. on Computing*, 27(1):291–301, 1998.
[4] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive online routing. *Proc. of FOCS*, 32–40, 1993.
[5] Y. Azar and O. Regev. Strongly polynomial algorithms for the unsplittable flow problem. *Proc. of IPCO*, 2001.
[6] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke. Optimal oblivious routing in polynomial time. *Proc. of STOC*, 2003.
[7] N. Bansal, A. Blum, S. Chawla and A. Meyerson. Online Oblivious Routing. *Proc. of SPAA*, 2003.

[8] M. Bienkowski, M. Korzeniowski, and H. Räcke. A Practical Algorithm for Constructing Oblivious Routing Schemes. *Proc. of SPAA*, 2003.

[9] A. Chakrabarti, C. Chekuri, A. Gupta, and A. Kumar. Approximation Algorithms for the Unsplittable Flow Problem. *Proc. of APPROX*, 2002.

[10] C. Chekuri and S. Khanna. On Multidimensional Packing Problems. To appear in *SIAM J. on Computing*. Preliminary version in *Proc. of SODA*, 185–194, 1999.

[11] C. Chekuri and S. Khanna. Edge Disjoint Paths Revisited. *Proc. of SODA*, 2003.

[12] C. Chekuri, M. Mydlarz and F.B. Shepherd. Multicommodity Demand Flow in a Tree. *Proc. of ICALP*, 2003.

[13] J. Chuzhoy and J. Naor. New inapproximability results for congestion minimization and machine scheduling. *Proc. of STOC*, 2004.

[14] Y. Dinitz, N. Garg and M. X. Goemans. On the single source unsplittable flow problem. *Combinatorica*, 19, 17–41, 1999. Preliminary version in *Proc. of FOCS*, 1998.

[15] S. Even, A. Itai and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. on Computing*, Vol 5, 691-703, 1976.

[16] A. Frank. Packing paths, cuts, and circuits - a survey. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, eds., *Paths, Flows and VLSI-Layout*, 49–100. Springer Verlag, Berlin, 1990.

[17] N. Garg, V. Vazirani and M. Yannakakis. Primal-Dual Approximation Algorithms for Integral Flow and Multicut in Trees. *Algorithmica*, 18(1):3-20, 1997. Preliminary version appeared in *Proc. of ICALP*, 1993.

[18] V. Guruswami, S. Khanna, R. Rajaraman, F. B. Shepherd, and M. Yannakakis. Near-Optimal Hardness Results and Approximation Algorithms for Edge-Disjoint Paths and Related Problems. To appear in *JCSS*. Preliminary version appeared in *Proc. of STOC*, 1999.

[19] C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. *Proc. of SPAA*, 2003.

[20] J. M. Kleinberg. Single-Source Unsplittable Flow. *Proc. of FOCS*, 1996.

[21] J. M. Kleinberg. Approximation algorithms for disjoint paths problems. PhD thesis, MIT, Cambridge, MA, May 1996.

[22] S. G. Kolliopoulos and C. Stein. Improved approximation algorithms for unsplittable flow problems. *Proc. of FOCS*, 426–435, 1997.

[23] P. Kolman and S. Scheideler. Improved bounds for the unsplittable flow problem. *Proc. of SODA*, 2002.

[24] P. Kolman. A Note on the Greedy Algorithm for the Unsplittable Flow Problem. *Information Processing Letters*, 88(3):101–105, 2003.

[25] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM*, 46(6):787–832, 1999. Preliminary version appeared in *Proc. of FOCS*, 1988.

[26] T. Leighton, S. Rao, and A. Srinivasan. Multicommodity flow and circuit switching. *Proc. of the Hawaii International Conference on System Sciences (HICSS)*, pages 459-465, 1998.

[27] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. Preliminary version in *Proc. of FOCS*, 1994.

[28] L. Lovász. *Combinatorial Problems and Exercises.* North-Holland, 1993.

[29] B. Maggs, G. Miller, O. Parekh, R. Ravi, and S. L. M. Wu. Solving symmetric diagonally-dominant systems by preconditioning. Manuscript, 2002.

[30] H. Räcke. Minimizing congestion in general networks. *Proc. of FOCS*, 2002.

[31] S. Plotkin. Competitive Routing of Virtual Circuits in ATM Networks. Survey in *IEEE Journal on Selected Areas in Communications*, 13(6):1128-1136, 1995.

[32] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[33] S. Rao. Small distortion and volume preserving embeddings for planar and Euclidean metrics. *Proc. of SoCG*, 300–306, 1999.

[34] N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver, Eds., *Paths, Flows and VLSI-Layout.* Springer-Verlag, Berlin, 1990.

[35] A. Srinivasan. Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. *Proc. of the FOCS*, pp. 416–425, 1997.

[36] K. Varadarajan and G. Venkataraman. Graph Decomposition and a Greedy Algorithm for Edge-disjoint Paths. In *Proc. of SODA*, 2004.