Meet and Merge: Approximation Algorithms for Confluent Flows

Jiangzhuo Chen [*] Rajmohan Rajaraman [*] Ravi Sundaram [†]

Abstract

In this paper we investigate the problem of determining *confluent* flows with minimum congestion. A flow of a given commodity is said to be confluent if at any node all the flow of the commodity departs along a single edge. Confluent flows appear in a variety of application areas ranging from wireless communications to evacuations; in fact, most flows in the Internet are confluent since Internet routing is destination based.

We consider the single commodity confluent flow problem, in which we are given an n-node directed network G, a sink t and supplies at each node, and the goal is to find a confluent flow that routes all the supplies to the sink while minimizing the maximum edge congestion. Our main result is an approximation algorithm, based on randomized rounding, for the special case when all the supplies are uniform; the algorithm finds a confluent flow with edge congestion $O(C^2 \log^3 n)$ where C is the node congestion of an optimal splittable flow. This implies an $\widetilde{O}(\sqrt{n})$ approximation algorithm for the problem. Our result relies on the analysis of a natural probabilistic process defined on directed acyclic graphs, that may be of independent interest.

For tree networks, we present an optimal polynomial-time algorithm for a multi-sink generalization of the above confluent flow problem. We show that it is NP-hard to approximate the congestion of the optimal confluent flow for general networks to within a factor of $4/3$. We also establish a lower bound on the gap between confluent and splittable flows, and consider multicommodity and fractional versions of confluent flow problems.

[*]College of Computer Science, Northeastern University, Boston MA 02115. Email:{chenj,rraj}@ccs.neu.edu. Partially supported by NSF CAREER award CCR-9983901.

[†]Akamai Technologies, Cambridge MA 02142. Email: koods@akamai.com.

1 Introduction

Consider the following three scenarios. *Scenario 1:* It is 5:30 AM on a bleak Saturday morning when a major fire engulfs a big hotel. The hotel guests and employees flee through the corridors and passageways as they follow the exit signs in their haste to get out. The streams of fleeing people meet and merge as they race towards the different emergency exits from which they pour out. *Scenario 2:* Content delivery networks (CDNs) employ vast deployments of servers distributed throughout the world to deliver content to users. For many applications ranging from distributing rich-media content to collecting billing data, CDNs often organize their deployment of servers in the form of a rooted tree (typically rooted at the Network Operations Center) with each node forwarding data from its children to its parent and vice versa. For *Scenario 3*, we turn to the wireless domain. As 802.11 networks get increasingly deployed (through airports and coffee shops) many start-ups are looking into setting up ad-hoc networks of Wi-Fi access points that act as forwarding agents back to the wired access point that connects to the Internet. An architecture that is commonly considered is for the ad-hoc network to be organized as a tree (rooted at the wired access point) with each Wi-Fi access point merge-forwarding the traffic from its children to its parent.

A common thread running through each of these scenarios is that all flows of a given commodity (be they people or CDN data or wireless packets) are *confluent*. A flow (of a given commodity) is said to be confluent[1] if all the flow (of this commodity) arriving at a node departs from the node along a single edge.

Confluent follows arise naturally in other applications as well. The biggest application, by far, is the Internet. Most flows on the Internet today are confluent because Internet routing is destination-based. Destination-based routing owes its widespread use and popularity to the fact that it saves memory on routers; requiring only one entry per node in a routing table, it leads to linear storage. Routing on the Internet is dictated by BGP which selects a shortest-path tree for each node at the Autonomous System level. BGP routing is destination-based. A major shortcoming of shortest-paths routing, however, is that it ignores congestion since the shortest paths are usually calculated independently for each source-destination pair.

In this paper, we study a number of optimization problems concerning confluent flows. Our primary interest is in finding confluent flows that minimize the maximum congestion in the network, the *congestion* of an edge (resp., node) being defined by the total amount of flow going through the edge (resp., node). We distinguish between the single commodity and multi-commodity variants. The bulk of this paper concerns the *single commodity confluent flow problem*, which is defined as follows. We are given a directed network, a distinguished sink, and supplies associated with each source. Our goal is to determine a confluent flow (that is a tree rooted at and directed toward the sink) such that the maximum congestion of any edge is minimized.

The maximum congestion of a confluent flow directed toward a single sink occurs at one of the edges incident into the sink. Thus, one can reduce the confluent flow problem into the following multi-sink problem. We are given a set of sinks t_1 through t_k and supplies at each node. We would like to determine a confluent flow that routes all the node supplies to the sinks such that the maximum flow arriving at any sink is minimized. We refer to this problem as the *single commodity multi-sink confluent flow problem*. In addition to being a useful reformulation, the multi-sink problem itself naturally captures several applications such as the evacuation scenario mentioned at the outset of this paper. Furthermore, while the single sink and multi-sink variants of the single commodity problem are equivalent for general networks, the multi-sink version is more general when considering certain special classes of networks (e.g., trees, meshes).

Our results. In this paper, we investigate the optimization and approximability of confluent flow problems.

- Our main result is for the single commodity problem for arbitrary networks with uniform supply at each node. We present an approximation algorithm, based on a simple randomized rounding, that

[1]The term "confluent flow" was suggested to us by Jon Kleinberg [15]; a similar notion of confluence (flows meeting and merging) also arose in some extensions that Kleinberg, Rabani, and Tardos were considering of their fair routing and load balancing algorithms [16].

determines a confluent flow with edge congestion $O(C^2 \log^3 n)$, where C is the node congestion of an optimal splittable flow. This implies an $\widetilde{O}(\sqrt{n})$-approximation algorithm for the single commodity flow problem, where n is the number of nodes in the network. We complement the preceding result by showing that the there exists an instance for which the randomized rounding algorithm incurs an $\Omega(n^{1/4})$ approximation ratio with probability $1-\varepsilon$, where $\varepsilon > 0$ can be made arbitrarily small. These results, which appear in Section 4, apply to both the single sink and multi-sink variants.

- We show that it is NP-hard to approximate the edge (or node) congestion of the optimal confluent flow in general directed networks to within a factor of 4/3, even for the uniform-supply case. We also show that there exists an instance for which the optimal splittable flow has constant edge and node congestion, while the best confluent flow has $\Omega(\log n)$ node congestion. These results may be found in Section 2.

- We provide optimal polynomial-time algorithms for the multi-sink single commodity problem (with arbitrary supplies) for both undirected and directed trees. The algorithms are described in Section 3.

- We also study the fractional and capacitated multicommodity variants of confluent flow problems. Our results for these problems, which appear in Section 5, include a proof that the randomized rounding procedure of Raghavan-Thompson [26] gives an $O(\log n)$ approximation for the multicommodity confluent flow problem, assuming the edge capacities are sufficiently large.

From a technical standpoint, perhaps the most challenging part of this work is the analysis of the randomized algorithm for single commodity confluent flow. Our result relies on the analysis of a natural probabilistic process defined on directed acyclic graphs, that may be of independent interest. While the particular process we study is closely related to branching processes and martingales, the bounds derived from existing general results in this area do not suffice for our purposes. Through a careful calculation of higher moments, we are able to derive useful bounds on the tail of the distribution of relevant random variables. (Due to space constraints, we have omitted many of the proofs; they may be found in the appendices.)

Related work. A natural question to ask is how does the congestion of the best confluent flow compare with that of the best splittable or unsplittable flow, both of which are natural relaxations of confluent flows. Splittable flows for the single commodity case are characterized by the well-known max-flow min-cut theorem of Ford and Fulkerson [8, 9]. The unsplittable flow problem, which requires that the supply from every source is routed along a single path, may be approximated to within a constant factor using the algorithms of [5, 17]. The relationship between the congestion of confluent and unsplittable flows is addressed in [22], in which an $\Omega(n)$ separation is established. Our results for confluent flow indicate that the optimal node congestion of (un)splittable flow is a better lower bound on the congestion of confluent flows. (See Section 2.)

To the best of our knowledge, the study of [22] is the first to explicitly discuss the effect of confluence on congestion in the context of IP routing. A recent study [27] explores the use of confluent flow-based routing (referred to as hop-by-hop routing) for premium traffic in the differentiated services framework of QoS routing. Both the studies [22] and [27] present heuristic solutions for various problems related to IP routing; these solutions, however, do not achieve non-trivial approximation factors. In [12, 19] confluent flows are also considered in a model where the demands are not associated with individual source-sink pairs; instead with sources or sinks, as a whole. Furthermore, the objective function being optimized is the total cost of edges used, not the maximum congestion. Also related is the work of [14], which raises the problem of finding a subtree of a given network that can route a given set of multicommodity flow pairs with minimum congestion.

Our results for the multicommodity version of confluent flow problems are straightforward generalizations of the rounding algorithm due to Raghavan and Thompson [26] for multicommodity unsplittable flows. Through a simple decomposition of fractional confluent flows, we are also able show that many of the existing results for concurrent fractional multicommodity flows (e.g., [20, 3, 21]) directly translate to corresponding results for concurrent fractional confluent flows.

The probabilistic process that we study in our analysis of the randomized rounding algorithm for general graphs, when restricted to trees, is similar to a random experiment analyzed for the group Steiner and Steiner covering problems [11, 18]. The measure of interest in our study is however complementary to that of [11, 18]. Our probabilistic process for trees can also be presented as a branching process ([7, Chapter XII], [2]) with different probabilities associated with each branch. Viewed in this context, our analysis bounds the upper tail of the distribution of the total progeny within a given number of generations. A related branching process is also analyzed in [6].

2 Preliminaries

In this section, we formally define the single commodity confluent flow problem. We defer to Section 5 for the definition of the multicommodity and fractional versions. Let $G = (V, E)$ be a directed network.

Definition 2.1 *Let $S \subseteq V$ be a set of sources and $t \in V$ be a distinguished sink. We say that a flow f from S to t is* confluent *if for every node u, there exists at most one edge (u, v) that has positive flow (i.e., $f(u, v) > 0$).*

In this paper, we seek confluent flows that have low congestion. One can consider two notions of congestion: *node congestion* and *edge congestion*. For a flow f, let the in-flow $f^{\text{in}}(v)$ (resp., out-flow $f^{\text{out}}(v)$) of a node v be $\sum_{(u,v)\in E} f(u, v)$ (resp., $\sum_{(v,u)\in E} f(v, u)$). Let *sup*$(v)$ denote the supply of node v. Note that for any node v, other than a sink node, we have $f^{\text{in}}(v) + \textit{sup}(v) = f^{\text{out}}(v)$. We define the *congestion* of node v as the larger of the in-flow and out-flow of the node and denote it by $f(v)$ for notational convenience. For a given flow f, the node congestion $NC(f)$ equals $\max_{v\in V-\{t\}} f(v)$, while the edge congestion $EC(f)$ equals $\max_{e\in E} f(e)$. (Note that the node congestion for a single commodity flow is given by the maximum congestion among the nodes, *excluding the sink*; the sink has the same congestion, regardless of the particular flow.)

Observation 1 *The edge congestion of a confluent flow is identical to its node-congestion.*

Proof: The edge congestion of a confluent flow is the flow on one of the edges incident into the sink t, say (u, t), and the node congestion is the flow through u, which is identical to $f(u, t)$. ■

Basic separation results. A major focus of this paper is the problem of finding a single commodity confluent flow that minimizes edge congestion. As mentioned in Section 1, Lorenz et al [22] present an instance (see Figure 1(a)) for which the edge congestion of an optimal confluent flow is $\Omega(n)$ times that of an optimal (un)splittable flow; their separation result holds even for the case when all nodes have unit supply. (Note that the integrality theorem for maximum flow implies that the congestions of the best unsplittable and splittable flows are equal.) We claim that the *node congestion* of the optimal splittable flow is a better lower bound on the edge (or node) congestion of the optimal confluent flow. Indeed, our result of Section 4 shows that this gap is $\widetilde{O}(\sqrt{n})$ for the special case of uniform supplies at each node.

In the following, we establish a lower bound of $\Omega(\log n)$ on the separation between (un)splittable and confluent flows, with respect to node congestion. Consider the instance of Figure 1(b). We have a k level complete binary tree, all the leaves of which are connected to a sink. Each node has unit supply. One splittable flow is given as follows. The supply from each node goes out on an arbitrary outgoing edge. Any flow coming into a node (at most 1 unit) is routed out on the edge not used by the supply at the node. This solution has edge congestion 1 and node congestion 2. But for any confluent flow, all supplies along path used by the supply at the root need to be routed through the same leaf. Since this path has length $k = \log(n)$, the edge and node congestions are at least $\log(n)$.

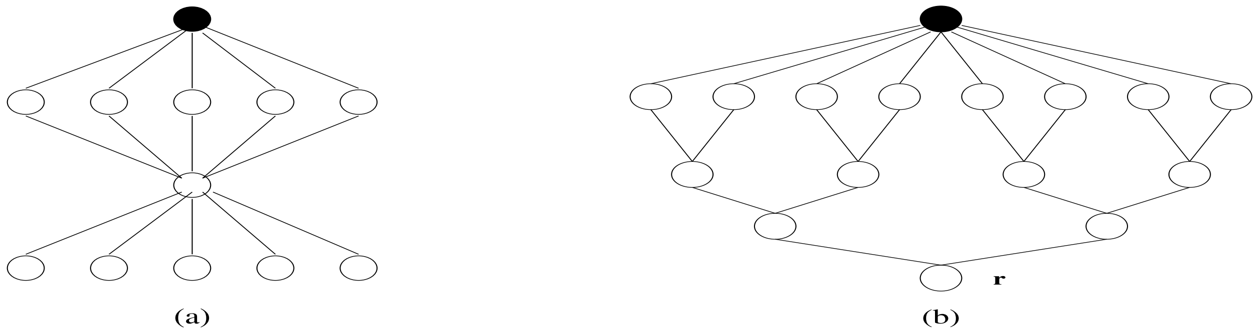

Figure 1: (a) An instance (due to [22]) which shows an $\Omega(n)$ gap between confluent and (un)splittable flows, with respect to edge congestion. (b) An instance which shows an $\Omega(\log n)$ gap between confluent and (un)splittable flows, with respect to node congestion. In each figure, all nodes offer unit supply and the dark node is the sink.

Hardness of approximation. We show that it is NP-hard to approximate the congestion of a confluent flow to a factor better than $4/3$, even for the special case when every node has unit supply. The reduction is from 2DIRPATH problem, which is shown to be NP-hard in [10] and defined as follows: Given a directed graph G and two node pairs s_1, t_1 and s_2, t_2, find node-disjoint directed paths from s_1 to t_1 and s_2 to t_2. We reduce this problem instance to an instance of the confluent flow problem as follows. We add a sink t with directed edges from t_1 and t_2 to t and attaching sources of size 1 to s_1 and t_2 and sources of size 2 to s_2 and t_1. We add a third node v with arcs from v to t and from all other nodes to v. We now add the following additional nodes and edges: (i) n^2 new nodes, each with an edge into s_1; (ii) n^2 nodes, each with an edge into t_2; (iii) $2n^2$ nodes, each with an edge to s_2; (iv) $2n^2$ nodes, each with an edge into t_1; and (iv) and a $3n^2$ nodes, each with an edge into v. It is easy to see that there exists a confluent flow with congestion at most $3n^2 + o(n^2)$ iff there exist node disjoint directed paths in the original graph; otherwise, the congestion is at least $4n^2$. Hence, the congestion of confluent flows is not approximable to better than $4/3$ in polynomial time unless 2DIRPATH is solvable in polynomial time.

Multi-sink problem. As discussed in Section 1, we solve the single commodity confluent flow problem by considering a more general multi-sink version of the problem, which is what we address in Sections 3 and 4. Given a set S of sources with supplies and a set T of sinks, a multi-sink confluent flow routes each supply to any sink such that the total flow is node-confluent. The congestion of a multi-sink confluent flow is the maximum congestion among all sinks of the network.

3 Multi-sink confluent flow for trees

Given an undirected tree $T = (V, E)$ with n nodes, each node v with nonnegative integer supply $sup(v)$ and k sinks $t_1, \ldots, t_k$, we would like to construct a multi-sink confluent flow from the sources to the sinks. By definition, the flow satisfies the following conditions: (i) all the supplies are routed to the sinks; (ii) the flow routing the supplies to a particular sink is confluent; (iii) the confluent flows to each sink are node-disjoint. The goal is to minimize the maximum congestion, which is simply the maximum total supply routed to any sink. Since the underlying network is a tree, our problem is equivalent to partitioning T into a set of subtrees such that there is at least one sink in each subtree and the maximal of the total supply, among all subtrees, is minimized. In this section, we present an optimal polynomial time algorithm for this problem.

Our algorithm, defined in Figure 2, consists of a binary search on the optimal congestion C in the range $[0, \sum_{v \in V} sup(v)]$ and uses a subroutine for finding whether there exists a multi-sink confluent flow with node congestion C; the subroutine also returns a flow with the desired property, should it exist. The intuition is simple: assign sources to the nearby sinks while they are not overloaded yet, and do this greedily.

Definitions.

- **Types of nodes:** A *sink-node* is a node that contains a sink; a *source-node* is a node that contains a source but no sinks; a *transit-node* is a node that is neither a source-node not a sink-node.
- **Parent and siblings:** For a leaf v, we define the *parent* of v, $p(v)$, to be the unique node adjacent to v and the siblings of v to be the leaves adjacent to the parent of v.
- **Load:** The *load* of a sink-node is the total supply of the sources located at the node.

TreeAlg(T): Perform binary search to determine the smallest C in $[0, \sum_{v \in V} \mathit{sup}(v)]$ for which the subroutine HASCONGESTION?(T, C) returns YES. Removing the edges marked by REMOVELEAF yields the desired confluent flow.

REMOVELEAF(v): Remove leaf v from the tree; if v is a sink-node, then mark the edge $(v, p(v))$.

MOVELEAF(v): Move all sinks in v, with their supplies, to $p(v)$, and remove v.

HASCONGESTION?(T, C): Repeat the following steps.

- If T has exactly one node, then return NO if the node is a source-node, and YES otherwise.
- Find a leaf of one of the following types and execute the corresponding operation.
  1. sink-node with a sink-node parent or transit-node: REMOVELEAF(v).
  2. source-node: If $\mathit{sup}(v) + \mathit{sup}(p(v)) \leq C$, then MOVELEAF($v$); otherwise, return NO.
  3. sink-node with a non-sink-node parent and some sink-node siblings: REMOVELEAF(w) for every sink-node leaf w adjacent to $p(v)$ except the one with minimum load.
  4. sink-node with a non-sink-node parent of degree exactly 2: if $\mathit{sup}(v) + \mathit{sup}(p(v)) \geq C$, then do REMOVELEAF($v$); otherwise, do MOVELEAF(v).

Figure 2: The algorithm for trees.

The proof of the correctness of HASCONGESTION? is in Appendix A. We also show in Appendix A how the algorithm can be adapted to apply for directed trees.

4 Multi-sink confluent flow for general networks

In this section, we present a simple randomized rounding algorithm for finding a multi-sink confluent flow in a given directed network. We analyze the algorithm for the case when every node has unit supply. Let t_1 through t_k denote the k sinks.

We first compute, using a standard maximum flow algorithm, a (splittable) flow f from the sources to the sinks t_1 through t_k such that the flow minimizes the maximum congestion among the sinks. This can be done by performing binary search on the congestion value, thus requiring $O(\log n)$ invocations of the maximum flow algorithm. We can assume, without loss of generality, that the edges used in the flow f constitute a directed acyclic graph since we can remove cycles without increasing the congestion on any node. In the following, we denote this dag by G.

We now round the flow f to obtain a confluent flow, which, by definition, is a forest of node-disjoint trees, each tree rooted at and directed toward a distinct t_i. The rounding procedure is simple. Recall that $f(e)$ denotes the flow on edge e and $f^{\text{out}}(v)$ denotes the flow out of node v. Each node v in $V - \{t_i : 1 \leq i \leq k\}$ independently selects exactly one of its outgoing edges, with the probability equal to $f(e)/f(v)$. (Note that the sum of the probabilities for all the outgoing edges of a node is 1.) The selected edges, together with the nodes in V, form a forest, each tree in the forest is an arborescence directed toward a distinct root t_i. Let T_i denote the tree rooted at node t_i. It is easy to see that the node-congestion of this confluent flow equals the maximum, over all i, of the number of nodes in T_i. The main result of this section is the following.

Theorem 1 *For the uniform-supply case, the congestion of the multi-sink confluent flow determined by* *randomized rounding is* $O((NC(f))^2 \log^3 n)$ *whp*[2].

Corollary 1.1 *The randomized rounding algorithm yields an* $\widetilde{O}(\sqrt{n})$ *approximation for the uniform-supply multi-sink node confluent flow problem.*

We analyze the randomized rounding algorithm by analyzing an equivalent probabilistic process $\mathcal{P}$ that is defined for any directed acyclic graph $\mathcal{D}$ with a *probability function* p on the edges of the dag, satisfying the condition that for every node u, $\sum_{(u,v)\in\mathcal{D}} p(u,v) \leq 1$. The process $\mathcal{P}(\mathcal{D})$ is simply the following: each node u selects at most one outgoing edge, the edge (u,v) selected with probability $p(u,v)$. The edges selected by $\mathcal{P}(\mathcal{D})$ form a forest of arborescences. Let $N_{\mathcal{D}}(v)$ denote the number of nodes in the subtree rooted at v in the forest. (Note that $N_{\mathcal{D}}(v)$ is a random variable.) We can calculate $C_{\mathcal{D}}(v) = E[N_{\mathcal{D}}(v)]$ as follows.

$$C_{\mathcal{D}}(v) = \begin{cases} 1 & v \text{ has no incoming edge} \\ 1 + \sum_{(u,v)\in\mathcal{D}} p(u,v) C_{\mathcal{D}}(u) & \text{otherwise.} \end{cases}$$

The randomized rounding algorithm is equivalent to the probabilistic process $\mathcal{P}(G)$ in which the probability function p is given by: $p(u,v) = f(u,v)/f^{\text{out}}(u)$. Furthermore, the congestion of the resultant confluent flow is the same as $C^*_{\mathcal{D}} = \max_v N_{\mathcal{D}}(v)$, while $NC(f)$ is the same as $\max_v C_{\mathcal{D}}(v)$. Thus, we can place a bound on the node congestion of the confluent flow determined by randomized rounding algorithm by bounding the tail of the distribution of the random variable $\max_v N_{\mathcal{D}}(v)$ for a given dag, which is what we set out to do in the following sections.

Our analysis of P proceeds in three steps. First, we show in Section 4.1 that for any dag $\mathcal{D}$, the height of every tree in the forest determined by $\mathcal{P}(\mathcal{D})$ is $O(C^*_{\mathcal{D}} \log n)$ whp. (We remark that this is the only step where we need the assumption about uniform supplies in the flow problem. In fact, our analysis can be generalized to bound the height in terms of the maximum ratio of the congestion and supply of a node.) Second, we show in Section 4.2 that all the moments of $N_G(v)$ are upper bounded by those of $N_{\mathcal{T}}(v)$ for an appropriate tree $\mathcal{T}$ obtained by unraveling the dag $\mathcal{D}$. In Section 4.3, we bound the moments of $N_T(v)$ for arbitrary trees T, which then yields the desired high probability bound on $N_{\mathcal{D}}(v)$. In Section 4.4, we put it all together and prove Theorem 1.

4.1 Bounding the height

We note that the distance between v and the root of the tree containing v in $\mathcal{P}(\mathcal{D})$ is the length of the random walk starting from v on $\mathcal{D}$, according to the probabilities defined by p. In the following, we omit the subscript $\mathcal{D}$ from the terms $C_{\mathcal{D}}(u)$ and $C^*_{\mathcal{D}}$ for notational convenience. For a given non-sink node u, let $P(u,i)$ denote the probability that the random walk is at node u after i steps. Let (w_1,u) through (w_j,u) denote the edges into u in $\mathcal{D}$. We then have the following recurrence relation:

$$P(u,i) = \begin{cases} 0 & i = 0 \text{ and } u \neq v \\ 1 & i = 0 \text{ and } u = v \\ \sum_{(w,u)\in\mathcal{D}} P(w,i-1)p(w,u) & i > 0. \end{cases}$$

Lemma 4.1 *For any non-sink node* u *and* $i \geq 0$, $P(u,i)$ *is at most* $C(u)(1-1/C^*)^i$.

Corollary 4.1.1 *The height of any tree in* $\mathcal{P}(G)$ *is* $O(C^* \log n)$ *whp.*

Proof: By Lemma 4.1, the probability that the random walk from v is at a non-sink node after $\alpha C^* \ln(nC^*)$ steps is at most $nC^*(1-1/C^*)^{\alpha C^* \ln(nC^*)} \leq \frac{1}{n^{\alpha-1}}$. Thus, the random walk from v terminates at a sink in $O(C^* \log n)$ steps whp, yielding the desired claim. ■

[2] We use the abbreviation "whp" throughout the paper to mean "with high probability" or, more precisely, "with probability $1 - n^{-c}$, where n is the number of nodes in the network and c is a constant that can be set arbitrarily large by appropriately adjusting other constants defined within the relevant context."

4.2 Reduction to the case of a tree

Consider the dag $\mathcal{D}$ with an associated probability function. We now argue that $\mathcal{P}(\mathcal{D})$ can be analyzed by considering the process $\mathcal{P}$ on an appropriately defined tree, obtained by a natural unraveling of G. We transform G into a tree $\mathcal{T}$ through a sequence of steps. Let D_j denote the dag obtained after j steps, $j \geq 0$. For a given directed acyclic graph D, let $D(v)$ denote the subgraph of D induced by all of the nodes that can reach v in G. (Note that $D(v)$ is also a dag.) Step $j+1$ proceeds as follows.

1. Find a node v in D_j such that the subgraph $D_j(v)$ is a tree and v has more than one incoming edge. If no such node is found, then D_j is a tree and the transformation is complete.

2. Let $(u_1, v), \ldots, (u_k, v)$ denote the $k \geq 2$ edges coming into v. We transform D_j into D_{j+1} as follows. We replace v and the subtree $D_j(v)$ rooted at v by k copies of each and replace the edge (u_i, v) by the edge (u_i, v_i), where v_i is the ith copy of v. Each new edge inherits the probability of the edge it replaces or copies. This is illustrated in Figure 4 in Appendix B.

Consider the jth step of the transformation $D_j \rightarrow D_{j+1}$. For a given node u and a nonnegative integer h, let the random variable $N_{D_j}(u, h)$ (resp., $N_{D_j}(u, h)$) denote the number of nodes within h hops of u in the subtree rooted at u under $\mathcal{P}(D_j)$ (resp., $\mathcal{P}(D_{j+1})$). We note that $E[N_{D_j}(u, h)]$ is equal to $E[N_{D_{j+1}}(u, h)]$. This implies the following equality.

$$C^*_{\mathcal{D}} = C^*_{\mathcal{T}} \tag{1}$$

While $N_{D_{j+1}}(u, h)$ has more "randomness" than $N_{D_j}(u, h)$ owing to multiple independent copies of the dag $D_j(v)$, it is not the case that the former random variable stochastically dominates the latter. As we show in Lemma 4.2, however, the moments of the former are at least that of the former.

Lemma 4.2 *For integers $h, r \geq 0$, and node u, we have $E[(N_{D_j}(u, h))^r] \leq E[(N_{D_{j+1}}(u, h))^r]$.*

The above lemma claims that for any copy of any node u, any moment of $N_{D_{j+1}}(u, h)$, for any height h, is no less than that of $(N_{D_j}(u, h))$. Since $\mathcal{D}$ is the first dag in the sequence and $\mathcal{T}$ is the last dag in the sequence, it follows that the moments of $N_{\mathcal{D}}(u, h)$ are bounded by those of $N_{\mathcal{T}}(u, h)$, for all u and h. In Section 4.3, we place a suitable upper bound on the moments of $N_T(u, h)$ for any tree T.

We now sketch a proof of Lemma 4.2. The process $\mathcal{P}(D_j)$ can be divided into two independent steps. The first step consists of the random choices of nodes not in $D_j(v)$ and the second step consists of the random choices of nodes in $D_j(v)$. Let K denote the set of nodes outside of $D_j(v)$ that are in the subtree rooted at u, following the first step. It is sufficient to show that given any K, the moments of $N_{D_j}(u, h)$ are bounded by that of $N_{D_{j+1}}(u, h)$, i.e.:

$$E[(N_{D_j}(u, h))^r \mid K] \leq E[(N_{D_{j+1}}(u, h))^r \mid K] \text{ for all } K \tag{2}$$

The desired claim then follows from Equation 2 and the fact that $E[W] = \sum_{Z=z} E[W|Z = z]Pr\{Z = z\}$ for all discrete random variables W and Z. The proof of Equation 2 follows from a tedious calculation using elementary algebra and is given in Appendix B.2.

4.3 Rounding on a tree

Sections 4.1 and 4.2 indicate that $N_{\mathcal{D}}(v)$, for any vertex v, can be bounded by placing an an upper bound on the moments of $N_{\mathcal{T}}(v, h)$ for $h = O(C^*_{\mathcal{D}} \log n)$. Let T be a tree rooted at node r with the edges directed toward the root, and with probability function p. In this section, we analyze $\mathcal{P}(T)$ and place a high probability bound on $N_T(r)$.

To simplify the analysis, we add a distinct leaf ℓ_v, for every node v in T, and an edge (ℓ_v, v) and set $p(\ell, v)$ to be 1. We also transform the resultant tree into a binary tree T' (i.e., every node has at most two children) by repeatedly applying the following transformation: replace every node with $k > 2$ children by two nodes, one of which has $\lfloor k/2 \rfloor$ children and the other has $\lceil k/2 \rceil$ children. Furthermore, we note that the number of leaves is identical in both the trees and the height of the binary tree is at most $\log n$ times that of the original tree. Let X_v denote the number of leaves that are in the subtree rooted at v obtained in process $\mathcal{P}(T')$. Clearly, $N_T(v)$ is identical to X_v, and $C_T(v) = E[X_v]$.

One technique for analyzing X_r is by using martingales (e.g., see [1, Chapter 7]). For instance, if the probability of each edge is $1/2$, then the sequence of variables corresponding to the number of nodes at each level of the tree that connect to the root forms a martingale. The bounds on the tail of X_r that we are able to get using martingale theory (e.g., the method of bounded differences [24]) are much weaker than what we want. Another technique, based on Janson's inequality, that has been used in analysis of a similar random experiment on trees [11, 18] yields lower bounds on X_r. Furthermore, our approach actually requires a bound on the moments of X_r, which we now establish in the following main lemma of this section. Let M_v denote the maximum $E[X_u]$ among all nodes u in $T'(v)$.

Lemma 4.3 *For any integer $i > 0$, we have $E[X_r^i] \leq \frac{i! M_r^{i-1} E[X_r] h^{i-1}}{2^{i-1}}$.*

Proof: The proof is by induction on the height h of the tree. We consider the base case $h = 1$, when r is a leaf. In this case, we have $E[X_r^i] = E[X_r]^i$. The right-hand side of the desired inequality is given by

$$\frac{i! E[X_r]^{i-1} E[X_r] h^{i-1}}{2^{i-1}} = \frac{i! E[X_r]^i}{2^{i-1}} \geq E[X_r]^i,$$

since $i! \geq 2^{i-1}$ for all integers $i > 0$.

We now consider the induction step. There are two cases depending on whether r has one child. We omit the case when r has one child since that is straightforward to prove. Consider the case when r has two children, say a and b. Let X_a and X_b denote the number of leaves in the subtrees, obtained after randomized rounding, containing a and b, respectively.

$$\begin{aligned}
E[X_r^i] &= p(a,r)(1-p(b,r))E[X_a^i] + (1-p(a,r))p(b,r)E[X_b^i] + p(a,r)p(b,r)E[(X_a+X_b)^i] \\
&= p(a,r)E[X_a^i] + p(b,r)E[X_b^i] + p(a,r)p(b,r)\sum_{j=1}^{i-1}\binom{i}{j}E[X_a^j]E[X_b^{i-j}] \\
&\leq p(a,r)E[X_a^i] + p(b,r)E[X_b^i] \\
&\quad + p(a,r)p(b,r)\sum_{j=1}^{i-1}\binom{i}{j}\frac{j! M_a^{j-1}E[X_a](h-1)^{j-1}}{2^{j-1}}\frac{(i-j)! M_b^{i-j-1}E[X_b](h-1)^{i-j-1}}{2^{i-j-1}} \\
&\leq p(a,r)E[X_a^i] + p(b,r)E[X_b^i] + p(a,r)p(b,r)E[X_a]E[X_b]\sum_{j=1}^{i-1}\frac{i! M_r^{i-2}(h-1)^{i-2}}{2^{i-2}} \\
&= p(a,r)E[X_a^i] + p(b,r)E[X_b^i] + p(a,r)p(b,r)E[X_a]E[X_b]\sum_{j=1}^{i-1}\frac{i! M_r^{i-2}E[X_a]E[X_b](h-1)^{i-2}}{2^{i-2}} \\
&\leq p(a,r)E[X_a^i] + p(b,r)E[X_b^i] + \sum_{j=1}^{i-1}\frac{i! M_r^{i-2}E[X_r]^2(h-1)^{i-2}}{4\cdot 2^{i-2}} \\
&\leq p(a,r)E[X_a^i] + p(b,r)E[X_b^i] + \sum_{j=1}^{i-1}\frac{i! M_r^{i-1}E[X_r](h-1)^{i-2}}{2^{i}} \\
&\leq p(a,r)\frac{i! M_a^{i-1}E[X_a](h-1)^{i-1}}{2^{i-1}} + p(b,r)\frac{i! M_b^{i-1}E[X_b](h-1)^{i-1}}{2^{i-1}}
\end{aligned}$$

$$
\begin{aligned}
& +\frac{(i-1)\cdot i!E[X_r]^{i-1}E[X_r](h-1)^{i-2}}{2^i} \\
= \;& \frac{i!M_r^{i-1}E[X_r](h-1)^{i-1}}{2^{i-1}} + \frac{(i-1)\cdot i!M_r^{i-1}E[X_r](h-1)^{i-2}}{2^i} \\
= \;& \frac{i!M_r^{i-1}E[X_r]}{2^{i-1}}\left((h-1)^{i-1} + (i-1)(h-1)^{i-2}\right) \\
\leq \;& \frac{i!M_r^{i-1}E[X_r]h^{i-1}}{2^{i-1}}.
\end{aligned}
$$

(In the seventh step, we use the fact that $p(a,r)p(b,r)E[X_a]E[X_b] \leq (p(a,r)E[X_a]+p(b,r)E[X_b])^2/4 = E[X_r]^2/4$. The eighth step follows from the inequality $E[X_r] \leq M_r$. The tenth step follows from the equation $p(a,r)E[X_a]+p(b,r)E[X_b] = E[X_r]$. In the last step, we use the inequality $(h-1)^{i-1} + (i-1)(h-1)^{i-2} \leq h^{i-1}$.) This completes the proof of the lemma. ■

Corollary 4.3.1 *For any $i \geq 0$, we have $E[(N_T(r,h))^i] \leq \frac{i!(C_T^*)^i h^{i-1}}{2^{i-1}}$.* ■

4.4 Putting it all together

In this section, we complete the proof of Theorem 1. Fix a sink t_j. By Corollary 4.1.1, we obtain that $N_G(t_j)$ equals $N_G(t_j,h)$ whp, where $h = O(C_G^* \log n)$. We now use Markov's inequality.

$$
\begin{aligned}
\Pr[N_G(t_j,h) > \alpha h C_G^*] &= \Pr[(N_G(t_j,h))^i > \alpha^i h^i (C_G^*)^i] \\
&\leq \frac{E[(N_G(t_j,h))^i]}{\alpha^i h^i (C_G^*)^i} \\
&\leq \frac{E[(N_T(t_j,h))^i]}{\alpha^i h^i (C_G^*)^i} \\
&\leq \frac{i!(C_G^*)^i h^{i-1}}{\alpha^i h^i (C_G^*)^i 2^{i-1}} \\
&\leq \frac{i!}{\alpha^i h 2^{i-1}}.
\end{aligned}
$$

(The second step follows from Markov's inequality. The third step follows from Lemma 4.2. The fourth step follows from Corollary 4.3.1 and Equation 1.) By setting $\alpha = i = O(\log n)$, noting that $NC(f) = C_G^*$, and taking into account the $O(\log n)$ factor due to the transformation to a binary tree, we obtain the desired claim of the theorem.

We note that our randomized rounding algorithm can be derandomized to yield a polynomial-time deterministic algorithm by the method of conditional probabilities and pessimistic estimators (see, for example, [1, Chapter 15] and [25]). The pessimistic estimator, in this derandomization process, is the sum, over all sinks, of the probability that the congestion at the sink exceeds the desired bound.

4.5 A lower bound on the randomized rounding algorithm

In this section we present an instance, with uniform supply at each node, for which the approximation ratio of our randomized rounding algorithm is $\Omega(n^{1/4})$. The instance is presented in Figure 3. The network has $n = \frac{3s^2+s}{2}$ nodes each with supply 1 and $2s-1$ sinks (on the boundaries). The optimal splittable flow consists of the flow splitting evenly at each node between the two outgoing (upward) edges. If the node is a sink (i.e. on the boundary) then half the flow is absorbed at the sink. If the node is the topmost sink then all the flow is absorbed. Note that this flow has maximum congestion $3s$ at the topmost sink. On the other hand, one can establish the following claim about the randomized rounding algorithm using basic analysis of a random walk on a line. With probability $1-\varepsilon$, for constant $\varepsilon > 0$, the route starting from the node c (at the center of the base) meeting the route starting from node v which is $\Theta(\sqrt{s})$ to the right of v. This implies an $\Omega(\sqrt{s})$ or $\Omega(n^{1/4})$ lower bound on the approximation ratio the randomized rounding algorithm. The full proof is in Appendix B.3.

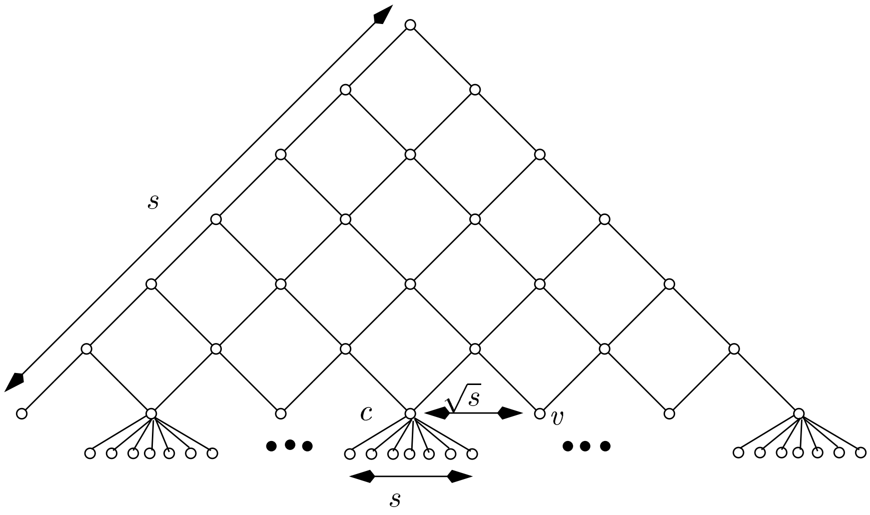

Figure 3: An instance for which the randomized rounding algorithm incurs an $\Omega(n^{1/4})$-approximation ratio.

5 Fractional and multicommodity confluent flows

In this section we consider the multicommodity confluent flow problem in which we are given a general directed graph, with one distinguished sink per commodity, arbitrary edge capacities, and arbitrary source supplies per commodity, and need to ensure that the flow per commodity is confluent. We begin, in Section 5.1, by studying a fractional version of the single- and multi-commodity confluent flow problem. The preceding analysis is then used in Section 5.2, where we consider integral multicommodity confluent flows.

5.1 Fractional confluent flows

A commonly-used metric in multicommodity flow problems is the *maximum concurrent flow* [20], which is defined the largest fraction of every commodity that can be shipped simultaneously without violating edge capacity restrictions. We extend this notion to single commodity confluent flows by defining the *maximum concurrent confluent flow* to be the largest fraction of each source supply that can be simultaneously routed to the sink using a confluent flow without violating edge capacity restrictions.

We note that the notion of concurrent flow is equivalent to the notion of *congestion ratio*, which is the maximum, over all edges, of the flow on the edge to its capacity. When all edge capacities are same, the congestion is the same as congestion ratio. Thus the integral version of the maximum concurrent confluent flow problem is equivalent to the minimum congestion problem presented in Section 2. In this section, we consider the fractional version of the maximum concurrent confluent flow problem. We begin by showing that any single commodity flow can be decomposed into a small number of fractional confluent flows.

Lemma 5.1 *Consider a flow f that routes supplies from several sources into a single sink. The flow f can be decomposed into a collection of at most m confluent flows, where m is the number of edges in the graph and where each confluent flow has the property that it satisfies the same fraction of every source's supply.*

It follows from Lemma 5.1 that any relationship between concurrent (non-confluent) flows and cuts also extends to fractional confluent flows and cuts. For example, one can extend the Ford-Fulkerson max-flow min-cut theorem to show the following. Define the *value of the sparsest cut* to be the minimum, over all cuts, of the ratio of the capacity of the cut to the total supply crossing the cut.

Corollary 5.1.1 *For a single commodity, the value of the sparsest cut is equal to the maximum concurrent flow achievable by fractional confluent flows.*

Similarly, for multiple commodities, one can extend the results of [3, 21] to show the following.

Corollary 5.1.2 *In the multicommodity case, the value of the sparsest cut is at most $O(\log n)$ times the maximum concurrent flow achievable by fractional confluent flows.*

5.2 Integral multicommodity confluent flows

Lemma 5.1, together with the Raghavan-Thompson rounding approach, yields an $O(\log n)$-approximation to the integral multicommodity confluent flow problem, assuming edge capacities are sufficiently large. We establish the claim in terms of congestion ratio; it implies an equivalent result for concurrent flow.

Theorem 2 *There exists an integral multicommodity confluent flow whose congestion ratio is no greater than $O(\log n)$ times that of the optimal (non-confluent) multi-commodity flow, provided the capacity of any edge in the graph is at least the total supply for any commodity.*

6 Discussion

In this section, we briefly discuss potential directions for future research and interesting open problems suggested by our work. We begin with considering another natural notion of confluence. Our definition of confluent flow captures *node-confluence*, since flows that meet at a *node* need to depart along the same edge. One can also define a notion of edge-confluence. We say that a flow f is *edge-confluent*, if for every node $u \in V - \{t\}$, there exists a mapping ϕ from the in-edges I_u of u to the out-edges O_u of u such that for all $e \in O_u$, $f(e) = \sum_{e':\phi(e')=e} f(e')$. It is easy to derive a polynomial-time approximation-preserving reduction from the confluent flow problem to the edge-confluent flow problem.

Two immediate problems left open by our work are to obtain better bounds on the gap between confluent flow congestion and splittable flow congestion and to bridge the gap between upper and lowe bounds on the approximation factor for arbitrary networks. A better, or more simplified, analysis of the randomized rounding algorithm for general networks would also be valuable. Given our polynomial-time optimal algorithm for trees, another potential line of research is to consider larger classes of networks such as planar networks, or meshes. Another interesting direction we are pursuing is to identify relationships among edge-confluent and confluent flows.

While our study is motivated in part by the applications of confluent flows in networks, there is considerable work that needs to be done before any of the solutions that we have presented can be migrated to practical scenarios. It will be interesting to explore the viability of finding confluent flows with limited information and in a distributed manner. Finally, bicriteria approximations in which multiple objective functions are considered (e.g., congestion and latency) are also topics worth pursuing (e.g., see [23]).

Acknowledgments

We would like to thank Jon Kleinberg, Aravind Srinivasan, and Éva Tardos for several helpful discussions.

References

[1] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley, New York, NY, 1991.

[2] K. Athreya and A. Vidyashankar. Branching processes. Technical Report TR–99–16, Department of Statistics, University of Georgia, 1999.

[3] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, 27(1):291–301, February 1998.

[4] H. Chernoff. A measure of the asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[5] Y. Dinitz, N. Garg, and M. Goemans. On the single-source unsplittable flow problem. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 290–299, November 1998.

[6] W. Evans, C. Kenyon, Y. Peres, and L Schulman. Broadcasting on trees and the ising model. *Annals of Applied Probability*, 10:410–433, 2000.

[7] W. Feller. *An Introduction to Probability Theory and its Applications*, volume 1. Wiley, New York, NY, 1967.

[8] L. Ford, Jr. and D. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[9] L. Ford, Jr. and D. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[10] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:111–121, 1980.

[11] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[12] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2001.

[13] W. Hoeffding. On the distribution of the number of successes in independent trials. *Annals of Mathematical Statistics*, 27:713–721, 1956.

[14] S. Khuller, B. Raghavachari, and N. Young. Designing multi-commodity flow trees. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, Nicola Santoro, and Sue Whitesides, editors, *Proceedings of the 3rd Workshop on Algorithms and Data Structures*, volume 709 of *Lecture Notes in Computer Science*, pages 433–441, Montréal, Canada, August 1993. Springer.

[15] J. Kleinberg, November 2002. Personal communication.

[16] J. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 568–578, October 1999.

[17] Jon M. Kleinberg. Single-source unsplittable flow. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 68–77, October 1996.

[18] G. Konjevod, R. Ravi, and A. Srinivasan. An approximation algorithm for the covering steiner problem. *Random Structures & Algorithms*, 20:465–48, 2002. Special Issue on Probabilistic Methods in Combinatorial Optimization.

[19] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener. Algorithms for provisioning virtual private networks in the hose model. In *Proceedings of the ACM SIGCOMM 2001 Conference*, volume 31 of *Computer Communication Review*, pages 135–148, August 2001.

[20] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *JACM: Journal of the ACM*, 46, 1999.

[21] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

[22] D. Lorenz, A. Orda, D. Raz, and Y. Shavitt. How good can IP routing be? Technical Report 2001-17, DIMACS, April 2001.

[23] M. Marathe, R. Ravi, R. Sundaram, S. Ravi, D. Rosenkrantz, and H. Hunt. Bicriteria network design problems. *Journal of Algorithms*, 28(1):142–171, July 1998.

[24] C. McDiarmid. On the method of bounded differences. In J. Siemons, editor, *Surveys in Combinatorics*, pages 148–188. Cambridge University Press, Cambridge, UK, 1989.

[25] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, Cambridge, UK, 2000.

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

[27] J. Wang and K. Nahrstedt. Hop-by-hop routing algorithms for premium-class traffic in diffserv networks. In *Proc. of IEEE INFOCOM 2002*, New York, NY, June 2002.

A Correctness of TreeAlg

It suffices to prove the correctness of HASCONGESTION?. Since the algorithm removes one leaf in every iteration and does not add any edges, the given network remains a tree. We next claim that the algorithm terminates. For this, we show that as long as the tree has at least two nodes, there exists at least one leaf satisfying one of the four conditions listed in the algorithm. A tree with at least two nodes has at least two leaves. If any of the leaves is a source-node or a transit-node (case 1 or 2), then our claim holds. Suppose all of the leaves are sink-nodes. If any of the at least two sink-node leaves has a sink-node parent (case 1), then again one of the conditions applies. Otherwise, one of the conditions holds if any two of the sink-nodes are siblings (case 3). Eliminating all of these possibilities leaves us with a tree with at least 4 nodes. Thus, all that remains to be shown is that when none of the above cases apply, there exists a leaf with a parent of degree exactly 2 (case 4). The following lemma establishes this claim.

Lemma A.1 *If any leaf of a tree with at least 4 vertices has no sibling, then there exist at least two parents with degree 2.*

Proof: Since a tree has at least 2 leaves, it has at least 2 parents. If they both have degree 2, we are done. Suppose parent v has degree 3. We travel along the internal vertices, starting from v. When an internal node:

(i). has degree 2 and is a parent, stop.

(ii). has degree 2 and is not a parent, continue on its unvisited neighbor.

(iii). has degree 3, continue on its unvisited non-leaf neighbor since it has at most one leaf neighbor.

Since it is a tree, the travel eventually stops and it stops on case (i). So we get one parent with degree 2. The travel has two options when starting at v. So there are at least 2 parents with degree 2. ■

Lemma A.2 *The algorithm* HASCONGESTION?*(T, C) returns* YES *iff tree T admits a multi-sink confluent flow of congestion at most C.*

Proof: It is sufficient to analyze a single iteration and show that if tree T^- is changed into T^+, T^+ has a YES answer if T^- has a YES answer, and T^- has a YES answer if T^+ has a YES answer. Suppose that T^- has a YES answer with a solution as partition P, and the iteration identifies a leaf v satisfying one of the desired conditions and executes the corresponding operation. Let us consider the various cases for v:

- Transit-node leaf: Any solution for T^+ is valid for T^-. Furthermore, no subtree in the solution for T^- needs to contain v.

- Sink-node leaf with a sink-node parent: Suppose P is a valid solution for T^-. If in P, v forms a subtree by itself, then P is also feasible in T^+. Otherwise, v is in the same subtree as its parent in P. But since the parent of v is also a sink-node, we can make v a separate one-node tree and the remaining partition still has a sink, which gives another YES solution to T^-. This partition is also valid in T^+. For the reverse direction, any valid solution for T^+, together with v as a separate tree, forms a valid solution for T^-.

- Source-node leaf: Suppose P is a valid solution for T^-. In P, v must be in the same subtree as its parent; so the sum of their supplies is at most C. So P, with v and its parent merged, is feasible in T^+; all subtrees still have the same total supplies as before. For the reverse direction, any valid solution for T^+, with the change that the node v is attached to its original parent, is also valid for T^-.

- Sink-node leaf with non-sink-node parent and sink-node siblings: Suppose P is a valid solution for T^-. If in P, the parent is disjoint with all these sink-node leaves, then P is feasible in T^+. If in P, the parent is joint with some of them, then another partition P' with all of them except the minimum-load one separated is also a YES solution to T^-. P' is feasible in T^+. Similarly, any valid solution for T^+ is also valid for T^-, after the addition of more subtrees, each consisting of one of the removed sink-nodes.

- Sink-node leaf with non-sink node parent of degree 2: Suppose P is a valid solution for T^-. We consider two cases. First, if $\mathit{sup}(v) + \mathit{sup}(p(v))$ is more than C, then in P, v and its parent cannot be in the same subtree; so P is feasible in T^+, even after removing v. Otherwise, if v and its parent are in the same subtree in P, then P is feasible in T^+. If they are separated, then we can cut the parent from its subtree and join it with v. The cut subtree still has a sink (since the parent is not a sink). The resulted partition is also a YES solution to T^- and is feasible in T^+.

  We now argue the other direction. Suppose P is a valid solution for T^+. If the operation in the iteration is the removal of v, then P, with one more subtree consisting of v alone, is a valid solution for T^-. Otherwise, P, with v attached to its original parent, is a valid solution for T^-.

■

The algorithm HASCONGESTION? can be modified to apply to directed trees. For a source-node leaf v, call its parent a *true parent* if the lone adjacent edge is going out of v, and a *false parent* otherwise. For a sink-node leaf v, call its parent a *true parent* if the lone adjacent edge is coming into v, and a *false parent* otherwise. As in HASCONGESTION? we seek a leaf that satisfies one of four conditions. Case 1 is the same as before. For case 2: we execute MOVELEAF only if the source-node leaf has a true parent and $\mathit{sup}(v) + \mathit{sup}(p(v))$ is at most C; otherwise, we return NO. Case 3 is the same as before. Finally, in case 4, we perform the same operations as for undirected trees if the sink-node leaf has a true parent; otherwise, we remove the leaf. We defer to the full paper the proof of correctness of the algorithm.

B Proofs for general networks

Proof of Corollary 1.1: If $NC(f)$ is at most $\sqrt{n}$, then the claim follows directly from Theorem 1. Otherwise, the claim trivially holds since the congestion of the confluent flow is at most n. ■

B.1 Proofs for bounding the height

Proof of Lemma 4.1: The proof is by induction on i. For $i = 0$, the claim is trivially true. For the induction step, we consider $P(u, i)$. By definition, we have

$$\begin{aligned}
P(u,i) &= \sum_{(w,u)\in\mathcal{D}} P(w,i-1)p(w,u) \\
&\leq (1-1/C^*)^{i-1} \sum_{(w,u)\in\mathcal{D}} p(w,u)C(w) \\
&\leq (1-1/C^*)^{i-1}(C(u)-1) \\
&\leq (1-1/C^*)^{i}C(u),
\end{aligned}$$

the last step following from the inequality $C(u) \leq C^*$. ■

B.2 Proof of Equation 2

We prove Equation (2) by induction on m. Suppose that v has m parents. In D_j, with probability $p_i,\ i \in M \equiv \{1, \ldots, m\}$, v chooses parent i; with probability $(1 - \sum_{i=1}^{m} p_i)$, v does not choose any outgoing edge. In D_{j+1}, each copy of v independently does the above. For simplifying notations, let X denote the random

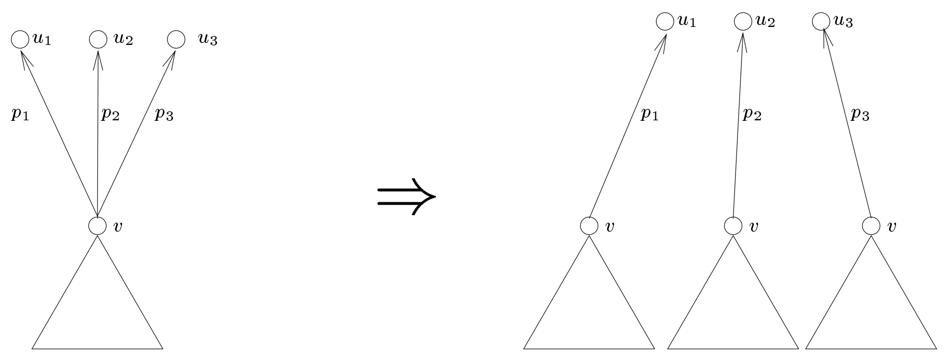

Figure 4: Transformation from dag to a tree, used in Section 4.2

variable $X(v)$. So in D_j, we have X. In D_{j+1}, we have $\{X_1, X_2, \ldots, X_m\}$ independently identically distributed (iid) as X. Now we can rewrite Equation (2) as:

$$\begin{aligned} LHS &= \sum_{i=1}^{m} E[(K+X)^r]p_i + K^r(1-\sum_{i=1}^{m} p_i) \\ RHS &= \sum_{\ell=0}^{m} \left(E[(K+\sum_{z=1}^{\ell} X_z)^r] \sum_{I=\{i_1,\ldots,i_\ell\}\subseteq M} \left(\prod_{i\in I} p_i \prod_{i\in(M-I)} (1-p_i) \right) \right) \end{aligned}$$

First we prove the following lemmas.

Lemma B.1 *For all non-negative random variables W, Z_1, Z_2, where Z_1 and Z_2 are independently identically distributed (iid) as Z:*

$$E[W^r] + E[(W+Z_1+Z_2)^r] \geq 2E[(W+Z)^r] \quad \forall r \geq 0$$

Proof:

$$\begin{aligned} E[W^r] + E[(W+Z_1+Z_2)^r] &= 2E[W^r] + \sum_{q=1}^{r} \binom{r}{q} E[W^{r-q}]E[(Z_1+Z_2)^q] \\ 2E[(W+Z)^r] &= 2E[W^r] + \sum_{q=1}^{r} \binom{r}{q} E[W^{r-q}]2E[Z^q] \end{aligned}$$

So the claim is true, since $E[(Z_1+Z_2)^q] \geq 2E[Z^q]\ \forall q \geq 0$. ■

Lemma B.2 *For all non-negative random variables $\{X_i | i = 1, 2, \ldots, \infty\}$ which are independently identically distributed (iid) as X, $\forall r \geq 0$:*

$$\begin{aligned} &E[(K+\sum_{z=1}^{\ell} X_z)^r](1-p_1-p_2) + E[(K+\sum_{z=1}^{\ell+1} X_z)^r](p_1+p_2) \\ &\leq E[(K+\sum_{z=1}^{\ell} X_z)^r](1-p_1)(1-p_2) + E[(K+\sum_{z=1}^{\ell+1} X_z)^r][p_1(1-p_2)+(1-p_1)p_2] \\ &+E[(K+\sum_{z=1}^{\ell+2} X_z)^r]p_1p_2 \end{aligned}$$

Proof: It immediately follows from Lemma B.1 (with $W = (K + \sum_{z=1}^{\ell} X_z)$, $Z_1 = X_{\ell+1}$ and $Z_2 = X_{\ell+2}$). ■

1. It is trivially true for $m = 0$ and $m = 1$.
2. For $m = 2$, it immediately follows from Lemma B.2 with $\ell = 0$.
3. Suppose $LHS \leq RHS$ for $m \geq 2$. Then for $m+1$, denote $p'_1 = p_1 + p_2, p'_2 = p_3, \ldots, p'_i = p_{i+1}, \ldots, p'_m = p_{m+1}$:

$$\begin{aligned}
LHS &= \sum_{i=1}^{m} E[(K+X)^r]p'_i + K^r(1 - \sum_{i=1}^{m} p'_i) \\
&\leq \sum_{\ell=0}^{m} \left(E[(K + \sum_{z=1}^{\ell} X_z)^r] \sum_{I=\{i_1,\ldots,i_\ell\}\subseteq M} \left(\prod_{i\in I} p'_i \prod_{i\in(M-I)} (1-p'_i) \right) \right) \\
&= \sum_{\ell=0}^{m-1} \sum_{I=\{i_1,\ldots,i_\ell\}\subseteq M:1\notin I} \left(E[(K + \sum_{z=1}^{\ell} X_z)^r] \prod_{i\in I} p'_i \left(\prod_{i\in(M-I-\{1\})} (1-p'_i) \right) (1-p'_1) \right. \\
&\quad \left. + E[(K + \sum_{z=1}^{\ell+1} X_z)^r] \left(\prod_{i\in I} p'_i \right) p'_1 \prod_{i\in(M-I-\{1\})} (1-p'_i) \right) \\
&= \sum_{\ell=0}^{m-1} \sum_{I=\{i_1,\ldots,i_\ell\}\subseteq M:1\notin I} \left(\left(\prod_{i\in I} p'_i \prod_{i\in(M-I-\{1\})} (1-p'_i) \right) \right. \\
&\quad \left. \left(E[(K + \sum_{z=1}^{\ell} X_z)^r](1-p_1-p_2) + E[(K + \sum_{z=1}^{\ell+1} X_z)^r](p_1+p_2) \right) \right) \\
&\leq \sum_{\ell=0}^{m-1} \sum_{I=\{i_1,\ldots,i_\ell\}\subseteq M:1\notin I} \left(\left(\prod_{i\in I} p_{i+1} \prod_{i\in(M-I-\{1\})} (1-p_{i+1}) \right) \right. \\
&\quad \left(E[(K + \sum_{z=1}^{\ell} X_z)^r](1-p_1)(1-p_2) + E[(K + \sum_{z=1}^{\ell+1} X_z)^r]p_1(1-p_2) \right. \\
&\quad \left. \left. + E[(K + \sum_{z=1}^{\ell+1} X_z)^r](1-p_1)p_2 + E[(K + \sum_{z=1}^{\ell+2} X_z)^r]p_1 p_2 \right) \right) \\
&= RHS
\end{aligned}$$

Note that the penultimate step follows from Lemma B.2.

B.3 Proof of lower bound for randomized rounding algorithm

Now consider the randomized rounding algorithm applied to this flow. Consider the node c in the center of the base and a node v that is horizontally at a distance $d = \Omega(\sqrt{s})$ to the right of c. The process of randomized rounding can be seen as proceeding level by level (starting at level 1) selecting an ancestor for each of c and v from the two potential candidates at each level. When the same ancestor is chosen it leads to the merging of the flows from v and c. Note that the merged flow has congestion $\Omega(s\sqrt{s})$. The distance between two ancestors at level i can be viewed as an unbiased random walk (with probability $\frac{1}{2}$ of remaining at the same point) that starts d steps to the right of the origin and the act of merging is equivalent to the random walk reaching the origin. Since the nearest sink node to c and v is more than distance $\Omega(s)$

away this random walk continues for at least $\Omega(s)$ steps. Further by throwing away all those levels at which the walk stays at the same point we are still left with $\Omega(s)$ steps at where the walk moves to the right or the left. This is a standard random walk on a line and the well-known results on the maxima and first passages of such random walks (e.g., see [7, Chapter III]), it follows that for any $\epsilon, 0 < \epsilon < 1$ there exists a δ such that the flows from c and v at distance $\delta\sqrt{s}$ merge with probability greater than $1 - \epsilon$. Hence with probability arbitrarily close to 1 the congestion achieved by randomized rounding is at least $\Omega(s\sqrt{s})$. Thus the ratio of the congestion in the confluent flow case to the splittable case is $\Omega(\sqrt{s})$ or $\Omega(n^{1/4})$.

C Proofs for fractional and multicommodity confluent flows

Proof of Lemma 5.1: Consider the graph induced by the edges with positive flow in f. Set the capacity of the edges to be the flow through that edge. Take any spanning tree on this graph and consider the largest possible fraction of each source supply that can be accommodated by this tree without violating edge capacity constraints. Since the underlying graph is a tree the flow is automatically confluent. Since the flow is maximal for the tree, it will bottleneck one or more edges; i.e., those edges will be at their maximum capacity. Remove this flow and consider the residual graph with zero capacity edges removed. The graph has fewer edges than the original graph. We repeat the above process and obtain a collection of at most m confluent flows, each flow supplying an equal fraction of each source supply, that collectively sum to the original flow f. ■

Proof of Theorem 2: Consider the optimal (non-confluent) multi-commodity flow obtained by solving the linear relaxation of the integral problem that allows fractional non-confluent flows. Let the congestion ratio achieved by this flow be C_{OPT}. By Lemma 5.1, this flow can be decomposed into at most m confluent flows per commodity. For commodity i, let D_i be the total supply and let an $r_{i,j}$ fraction of D_i be routed by the jth confluent flow i.e $\Sigma_j r_{i,j} = 1$. Consider the algorithm that randomly selects one confluent flow for each commodity i, picking the jth flow with probability $r_{i,j}$, and scales up that flow to fraction 1, i.e. that flow now has value D_i. It is obvious that this collection of flows, one per commodity, routes the supplies for all commodities and is confluent.

We now analyze the flow through any edge e with capacity c_e. Let r_i^e be the fraction of commodity i flowing through this edge in the optimal non-confluent flow. By definition $\Sigma_i r_i^e D_i \leq c_e \times C_{\text{OPT}}$. Then the value of the flow through this edge in the confluent case is the sum of independent random variables X_i where X_i is D_i with probability r_i^e and 0 otherwise. The expected value of this sum is $\Sigma_i r_i^e \times D_i \leq c_e \times C_{\text{OPT}}$. Let $D_{\max}$ be the largest supply of any commodity and let $Y_i = X_i / D_{\max}$. Then $E(\Sigma_i Y_i) \leq c_e \times C_{\text{OPT}} / D_{\max}$.

We now apply Chernoff-Hoeffding bounds [4, 13] by considering two mutually exclusive but exhaustive cases. If $E(\Sigma_i Y_i) \leq 24 \log n$, then we obtain that $\Pr(\Sigma_i Y_i > 48e \log n) < n^{-48e}$ Otherwise, we obtain that $\Pr[\Sigma_i Y_i > 1.5E(\Sigma_i Y_i)] < n^{-2}$. The two cases together yield that with high probability $\Sigma_i Y_i < \max\{O(c_e \times C_{\text{OPT}}/D_{\max}), O(\log n)\}$,i.e. $\Sigma_i X_i < \max\{O(c_e \times C_{\text{OPT}}/), O(D_{\max} \log n)\}$. Taking the union bound over all edges and using the fact that $c_e \geq D_{\max}$ we see that there exists a selection of confluent flows one per commodity so that the capacity of no edge is violated by more than $c_e \cdot O(C_{\text{OPT}}, \log n)$. Hence there exists a confluent flow with congestion ratio at most $O(\log n)$ times the congestion ratio of any (non-confluent) multicommodity flow. This randomized scheme can be made deterministic by the method of conditional probabilities and pessimistic estimators (see, for example, [1, Chapter 15] and [25]) in a straightforward fashion. (The pessimistic estimator, in our case, is the sum, over all edges, of the probability that the congestion ratio of the edge exceeds the desired bound.) ■