Upward Max Min Fairness

Emilie Danna[§], Avinatan Hassidim[*], Haim Kaplan[*†], Alok Kumar[§], Yishay Mansour[*†], Danny Raz[*‡], Michal Segalov[*]
[*]Google, Inc. Israel R&D Center
[§]Google, Inc. Mountain View, USA
[†]Tel Aviv University
[‡]Technion, Israel
{edanna, avinatan, haimk, kumaralok, mansour, msegalov, razdan}@google.com

***Abstract*—Often one would like to allocate shared resources in a fair way. A common and well studied notion of fairness is *Max-Min Fairness*, where we first maximize the smallest allocation, and subject to that the second smallest, and so on. We consider a networking application where multiple commodities compete over the capacity of a network. In our setting each commodity has multiple possible paths to route its demand (for example, a network using MPLS tunneling). In this setting, the only known way of finding a max-min fair allocation requires an iterative solution of multiple linear programs. Such an approach, although polynomial time, scales badly with the size of the network, the number of demands, and the number of paths. More importantly, a network operator has limited control and understanding of the inner working of the algorithm. Finally, this approach is inherently centralized and cannot be implemented via a distributed protocol.**

In this paper we introduce Upward Max-Min Fairness, a novel relaxation of Max-Min Fairness and present a family of simple dynamics that converge to it. These dynamics can be implemented in a distributed manner. Moreover, we present an efficient combinatorial algorithm for finding an upward max-min fair allocation. This algorithm is a natural extension of the well known Water Filling Algorithm for a multiple path setting.

We test the expected behavior of this new algorithm and show that on realistic networks upward max-min fair allocations are comparable to the max-min fair allocations both in fairness and in network utilization.

I. Introduction

The allocation of global shared resources to different users is a fundamental problem in distributed computing and networking. A well accepted hypothesis is that network resources belong to the community and thus should be shared in a fair way among all users. This is considered by many to be the basic philosophy behind the congestion control mechanism of TCP, which is one of the most important technical building blocks of the Internet and a major contributor to its success.

In general, the notion of fairness is wide and covers many specific allocation strategies. When there is only one resource from which all users can benefit equally, then a fair allocation will allocate an equal portion of the resource to each user (up to its demand). However, in many realistic scenarios, the situation is more complex. Consider for example a traffic engineering setting where the goal is to route traffic of many commodities using the available (shared) network capacity. In this case, providing each commodity with an equal share of the capacity of each link does not make sense anymore. Such allocation will cause substantial waste of resources since the amount of flow that a commodity can send along a path p is determined by its smallest allocation on an edge along p.

The goal is to allocate the common resources (such as network capacity) in a fair way while utilizing them as much as possible. This gives rise to what is known as the "Water Filling" algorithm (Waterfill) [3]. Assume that each commodity has a single path connecting its source to its destination, which is indeed the case in networking settings where the routing is determined by the network layer (IP) protocol, see e.g., [10]. When we start all commodities are active. We increase the flow of all commodities equally until the first link (or links) in the network becomes saturated.[1] Then, all commodities whose path contains this saturated link cannot utilize any additional allocation and we deactivate them. We continue increasing all active commodities equally until another link (or a set of links) gets saturated, deactivate the relevant commodities and continue until no active commodity remain.

The outcome of the Waterfill algorithm, in the single path traffic engineering setting, is what is known as a *max-min* fair solution (max-min fair multicommodity flow in our case) [3]. It is not hard to see that in such a max-min fair multicommodity flow we cannot increase the flow of any commodity without decreasing the flow of commodities with equal or less flow. In fact, it is common to define an allocation to be max-min fair if in order to increase the amount allocated to one user we have to decrease the amount allocated to a user which gets an equal or less amount. In other words, a multicommodity flow is max-min fair if the sorted vector of flow values is the lexicographically largest feasible sorted vector of flow values. This vector is known to be unique [3].

This centralized Waterfill algorithm can be replaced by simple distributed dynamic in which each commodity checks if it can increase its value while only decreasing the values of commodities with strictly larger values. If this is indeed the case, the dynamic performs the change. This simple dynamic is guaranteed to converge to the max-min fair solution [1].

When traffic can be sent along multiple paths between the source and the destination, the situation become much more complex. This is the case in many practical traffic engineering scenarios, where ISPs are using multiple paths (for example

[1]A link is saturated if the total flow through it equals its capacity

using MPLS tunneling) to achieve better load balancing and to increase the overall throughput and utilization of their networks. In such a setting, the vector of flow values is no longer sufficient to uniquely describe the load on each link since the flow of commodity i can be split in different ways among the different paths that commodity i uses. However, even when we have multiple paths we can still define a max-min fair multicommodity flow to be one with the largest lexicographically sorted vector of flow values among all sorted vectors of flow values of feasible flows. This vector is unique, but there may be more than one multicommodity flow that is represented by this vector.

One can extend the distributed dynamic we described earlier to the case when there is more than one path per commodity. In this case, we will allow a commodity to increase its flow value along a path p, if it can do so by decreasing only commodities of strictly larger flow value, along paths p' intersecting p. Note that we compare the total value of a commodity (over all its paths) rather than its value on the paths p' intersecting p. It is clear that each such an update operation locally improves fairness. There are two basic intriguing questions regarding this dynamic for the multiple paths setting:
(1) Does the dynamic always converge? and
(2) In case it does converge, does it converge to a max-min fair allocation ?
In this paper we show that, under mild assumptions, such a dynamic always converges, but unfortunately not necessarily to a max-min fair multicommodity flow. The dynamic converges to a fairness notion which we call *Upward Max-Min Fair (UMMF)*, which has many intuitive fairness guarantees.

A. Our contribution

I. UMMF - A new notion of fairness. We define a multicommodity flow to be Upward Max-Min Fair (UMMF), if for each i, we cannot increase the value of the ith smallest commodity, along any of its paths, even if we remove all commodities whose value is strictly larger than the ith smallest value. (This implies that if we reach an UMMF then the dynamic described earlier terminates.)

At a first glance this definition may look like the traditional definition of max-min fair multicommodity flow stated in terms of flows rather than flow values. However, there is a very important subtle difference between the two definitions. In UMMF the requirement is that we cannot increase the ith largest commodity while fixing the current flow routing of the smaller or equal commodities. In the traditional definition the requirement is stronger: the flow is max-min fair if we cannot increase the ith largest flow value for any allocation that achieves the maximal values for the smaller or equal commodities. For example, if we can reroute the commodities with value smaller than or equal the ith largest so that the ith largest commodity increases then the flow would not be max-min fair but it may be upward man-min fair. It is clear that any max-min fair multicommodity flow is also upward max-min fair multicommodity flow, but not vice versa. To better distinguish between the two notions we shall refer to the traditional stronger notion as a Global Max-Min Fair (GMMF) multicommodity flow.

The example in Figure 1 highlights the differences between UMMF and GMMF. In this example the only global max-min fair flow is the one with $(1, 1)$ as the vector of sorted values. On the other hand, there are many different UMMF flows with different vectors of sorted values. In fact, for every $0 \leq \alpha \leq 0.5$ if flow i (for $i \in \{1, 2\}$) routes α flow units on the path s_iABCt_i and $1 - 2\alpha$ flow units on the other path, then the resulting multicommodity flow is UMMF. This holds since the flow is maximal (no flow on a single path can be increased) and both commodities have the same flow value. The example can be extended to show that the UMMF can have a total flow which is a factor of $O(n)$ from the GMMF, where n is the number of nodes.

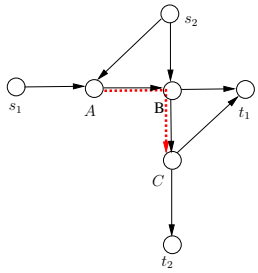


Fig. 1. In this example $P_1 = \{(s_1ABt_1), (s_1ABCt_1)\}$ and $P_2 = \{(s_2, BCt_2), (s_2ABCt_2)\}$

As indicated by this simple example, a UMMF multicommodity flow is not unique. However, every UMMF flow is "fair" in a sense similar to the one by which GMMF multicommodity flows are fair: to increase a commodity along a path we have to change the allocation of equal or smaller commodities. To further argue that UMMF multicommodity flow is a natural concept, consider the case where there is only one commodity. In this case any maximal flow (sometimes also called blocking flow) is UMMF.[2] On the other hand only a maximum flow is GMMF. The notion of blocking flow is well known, and can be computed faster than a maximum flow (in fact computing a blocking flow is an essential step in Dinic's maximum flow algorithm. [4]).

II. Distributed Dynamics. We give a set of simple conditions such that any dynamics obeying these conditions converge to a UMMF multicommodity flow. At a high level all one has to do is to pick a commodity and increase its allocation while decreasing the allocation of larger commodities. If we keep the value of the commodity that increases smaller than the value of those that decrease then we prevent oscillations and guarantee convergence.

III. Generalized Waterfill Algorithm. The Waterfill algorithm described earlier is a classical simple algorithm to compute the max-min fair multicommodity flow in the case of a single path per commodity.

[2]A blocking flow is a flow which we cannot increase with rerouting some part of it.

We suggest a natural extension of the Waterfill algorithm, *Iterative Exhaustive WaterFill (IEWF)*, for the case of multiple paths per commodity and show that it converges to an UMMF multicommodity flow. This, in fact, reinforces our claim that the notion of UMMF multicommodity flow is a natural one.

In addition to the the fact that an UMMF flow can be computed in a simple distributed manner, there is also an important computational benefit gained by the introduction of UMMF and the associated dynamics. Unfortunately, there is no combinatorial algorithm to find a max-min fair solution when there are multiple possible paths per commodity. Finding one is a challenging open problem. Still, it is possible to find a max-min fair flow by solving multiple linear programs, at least one to discover every flow value [8]. Although it runs in polynomial time, it scales badly with the size of the network, the number of demands, and the number of paths (all cause the LP to grow). Furthermore, a network operator has limited control and understanding of the inner working of the algorithm, and the network has to switch from one allocation to another allocation in a synchronized way. Such an implementation may also be unstable in the sense that changing slightly the capacities or the demands may have a large affect on the solution.

III. Experimental study. We performed an experimental study to check various properties of UMMF multicommodity flows and the IEWF algorithm: We conducted our tests on the Google backbone network and on random networks drawn using Waxman's model [11], [12]. We compared the GMMF multicommodity flow to a UMMF multicommodity flow obtained after 2, 10, and 50 iterations of IEWF. All our measurements indicate that the quality, in terms of fairness and throughput, of the UMMF flow which we obtain (even after 2 iterations and definitely after 10) is close to the quality of the GMMF flow.

We investigated the affect of the initial splits on the quality of the UMMF flow and the rate of convergence. We compared several natural choices of initial splits and indicate the ones that give the best performance. We also compared how sensitive are the GMMF and UMMF multicommodity flows to small changes of the inputs. Our results show that the UMMF flow is much more robust.

II. Upward Max-Min Fair Flow

We are given a directed graph $G = (V, E)$, where each edge $e \in E$ has a capacity constraint $c(e) > 0$.

A multicommodity flow problem, has k tuples (s_i, t_i, P_i), for $1 \leq i \leq k$, one per commodity, where s_i is the source of commodity i, t_i is the destination of commodity i, and P_i is a subset of simple paths from s_i to t_i which commodity i can use. (Note that P_i can be potentially all paths connecting s_i to t_i.)

A *flow* f_i of commodity i associates a value $f_i(p) \geq 0$ with each path $p \in P_i$. The value of f_i is $v(f_i) = \sum_{p \in P_i} f_i(p)$. A *multicommodity flow* F is a vector $(f_1, f_2, \ldots, f_k)$ of flows f_i, $1 \leq i \leq k$. The multicommodity flow is *feasible* if for every edge $e \in E$, we have $\sum_i \sum_{p \in P_i} f_i(p) \leq c(e)$.

Consider a multicommodity flow $H = (h_1, \ldots, h_k)$. Let $\sigma(i)$ be the flow with the ith smallest flow value (we assume that we break ties by the name of the flow unless stated otherwise), That is $v(h_{\sigma(i)}) \leq v(h_{\sigma(i+1)})$, for $1 \leq i \leq k-1$. Let $v(H) = (v(h_{\sigma(1)}), v(h_{\sigma(2)}), \ldots, v(h_{\sigma(k)}))$ the vector of flow values in non-decreasing order. We define the *rank* of commodity j to be $\sigma^{-1}(j)$: This is the position of commodity j in the sorted sequence of the commodities by their values.

To simplify notation we shall assume in the rest of this section that $\sigma(i) = i$, that is the commodities in H are indexed such that $v(h_1) \leq v(h_2) \leq \cdots v(h_k)$.

Definition II.1 (UMMF)**.** *Let F be a feasible multicommodity flow. For every $1 \leq i \leq k$ define $L_i = \{f_j \mid v(f_j) > v(f_i)\}$. We say that F is* Upward Max-Min Fair *if for every flow i, even after we delete all flows in L_i, the flow values $f_i(p)$ for $p \in P_i$, are maximal.*

Note that in case where we have only one commodity then a flow is UMMF if and only if it is a maximal flow. (I.e., it does not have to be a maximum flow.) The following claim gives an alternative formulation of UMMF.

Claim II.2. *A feasible multicommodity flow F is UMMF if and only if for every flow f_i and every path $p \in P_i$, there exists an edge $e \in p$ such that e is saturated only by commodities with value not larger than $v(f_i)$. Specifically, F is UMMF if and only if*

$$\forall i \forall p \in P_i \exists e \in p \sum_{p' \mid (e \in p') \wedge (p' \in P_j) \wedge (v(f_j) \leq v(f_i))} f_j(p') = c(e) \ .$$

III. The Lexicographic-Increase Dynamics

A. Overview

In this section we consider dynamics in which at each step one commodity is increased along one of its paths, at the expense of commodities which have a higher flow value. We require that the increase is bounded, such that the value of the increased commodity (after the increase) is upper bounded by the value of each decreased commodity (after the decrease). We prove that any such dynamics converges to a limit, and that any limit of such dynamics is UMMF.

Consider first the trivial case in which the sequence has only increase operations. In this case clearly the sequence has a limit, since the flow values are bounded. Unfortunately, flow values can both increase and decrease. So the main challenge is to handle the decrease operations, and to bound them.

The proof focuses on the sorted vector of flow values $v(H) = (v(h_{\sigma(1)}, \ldots, h_{\sigma(k)})$ for a multicommodity flow H, defined in Section II. Note that the identity of $\sigma(i)$, the ith largest flow, changes over time as the dynamic progresses. Consider the minimal flow value $v(h_{\sigma(1)})$. The identity of the commodity with the smallest flow value may change over time, but the smallest flow value can never decrease, given our definition of the dynamics. Now consider the second smallest flow value. This value might decrease in certain time steps. However, by the definition of the dynamics, each time it decreases, the smallest value has to increase. The main

challenge in the convergence proof is to charge the decreases of flow values of large indices to increases of flow values of smaller indices.

The first step in the proof is to split every sequence of steps (which respects the dynamic) to basic operations of modifying a single commodity on a single path, either decreasing or increasing it. Any step of the dynamics is split to a single increase and potentially multiple decreases. To simplify the proof, we require that the increase is done first, and then the decreases. This implies that the intermediate steps might not represent a feasible flow. However, we show that if there is an infinite sub-sequence which is feasible, then the limit is a feasible flow.

For the proof we define the notion of a *happy sequence*. The main property of the happy sequence is that it will allow us to associate each decrease with an increase that occurred before it, such that the increase was to a commodity with a lower rank (recall that the *rank* of a commodity j is $\sigma^{-1}(j)$).

B. Definitions

Consider a sequence H^1, H^2, ... of multicommodity flows. We call a sequence *happy* if it satisfies the following conditions:

1) The value of each commodity is bounded by R, for $R > 0$.
2) The difference between H^s and H^{s+1} is only in the amount of flow along a single path of a single commodity. Moreover, this does not change the order between commodities (up to tie breaking). Formally, let $1, \dots k$ denote the commodities. For every s, we require that there exists a permutation σ_s such that for ever i we have $v(h^s_{\sigma_s(i)}) \leq v(h^s_{\sigma_s(i+1)})$ and $v(h^{s+1}_{\sigma_s(i)}) \leq v(h^{s+1}_{\sigma_s(i+1)})$.
3) There is a function Π that maps indices in the sequence to smaller indices defined as follows. If at step s we decrease the flow of commodity i along a path, then at step $\Pi(s) < s$ we increase the flow of commodity i' along a path. Furthermore, if the rank of commodity i in H^s is j, and the rank of commodity i' in $H^{\Pi(s)}$ is j', then $j' < j$.
4) Let $decrease(s)$ or $increase(s)$ be the amount by which we decrease or increase the flow along a path at step s, respectively. For any step s' in which we increase the flow along a path let $\Pi^{-1}(s') = \{s \mid \Pi(s) = s'\}$. Then we have $m \cdot increase(s') \geq \sum_{s \in \Pi^{-1}(s')} decrease(s)$, where m is the number of edges in the graph.

C. Happy Sequence converges

Theorem III.1. *If H^s is a* happy *sequence, then $v(H^s)$ has a limit.*

Proof: Recall that $\sigma(i)$ is the commodity with the ith smallest value. Consider $v(h^s_{\sigma(1)})$. This value is non-decreasing throughout the dynamics and is bounded by R so it must have a limit. Recall that the identity of the commodity $\sigma(1)$ may change throughout the process but the value $v(h^s_{\sigma(1)})$ does not decrease.

We show that the sum of the decreases of $v(h^s_{\sigma(2)})$ is bounded: Since the total increase of $v(h^s_{\sigma(1)})$ is at most R, the sum of the decreases of $v(h^s_{\sigma(2)})$ is at most mR. Furthermore, since $v(h^s_{\sigma(2)})$ is also bounded by R, the sum of the increases of $v(h^s_{\sigma(2)})$ is at most $R+mR$. It follows that both the sum of increases and the sum of the decreases of $v(h^s_{\sigma(2)})$ are bounded and therefore have a limit and therefore $v(h^s_{\sigma(2)})$ has a limit.

In general let A_i be an upper bound on the sum of the increases of $v(h^s_{\sigma(i)})$. Then we have that $A_i \leq m(A_1 + A_2 + \dots + A_{i-1}) + R$. Solving this recurrence we get that $A_i \leq R(m+1)^{i-1}$. It follows that the sum of the decreases and the sum of the increases of $v(h^s_{\sigma(i)})$ are bounded and therefore they have a limit and $v(h^s_{\sigma(i)})$ has a limit. ■

Theorem III.1 shows that the vector of the sorted values of the commodities converges. But what about the multicommodity flow itself? We represent the multicommodity flow H itself as a vector of the flow on each path, and denote this vector $p(H)$. Each coordinate of $p(H)$ corresponds to a path p of some commodity and the value of the coordinate is the amount of flow that the particular commodity sends along p. For a *happy* sequence of multicommodity flows H^s the difference between $p(H^s)$ and $p(H^{s+1})$ is in exactly one coordinate. (Following proof is omitted due to lack of space.)

Corollary III.2. *If H^s is* happy *then the sequence $p(H^s)$ has a limit corresponding to some flow H^*. In addition, if H^s is* happy *and it has a infinite subsequence that is a feasible flow then it has a limit, H^*, which is a feasible flow.*

D. From a happy sequence to dynamics

Consider dynamics in which as long as we have a flow H which is not UMMF we pick some commodity h_i, increase its value along some of its paths while decreasing the flow along paths of commodities h_j such that $v(h_j) > v(h_i)$. Let h'_i denote commodity i after the increase and let h'_j denote commodity j after the decrease. We guarantee that

1) For every path p of commodity h_i the value that h_i sends along p does not decrease, and for every path p of h_j, the value of that h_j sends along p does not increase.
2) $v(h'_j) \geq v(h'_i)$.
3) If h_i increases by Δ then we decrease any h_j by at most $m\Delta$.

Let H^s be the sequence of multicommodity flows generated by such a dynamics. It is easy to verify that by refining the transition from H^s to H^{s+1} into multiple smaller steps we can obtain a *happy* sequence. Therefore by Corollary IV-C the sequence H^s converges to a feasible flow H^*.

How do we guarantee that the limit flow H^* is UMMF? For that we add the following two additional requirements:

4) If a commodity can increase while changing only larger commodities, then in a finite number of steps we will either increase it or it will not be possible to increase it anymore.
5) When we increase a commodity then we do it by at least a constant fraction ζ of its maximum possible increase (subject to the conditions of the dynamics).

The following theorem (proof omitted) derives the convergence.

Theorem III.3. *For any dynamic satisfying Conditions (1)-(5) above the sequence of multicommodity flow H^s converges to a feasible multicommodity flow H^* which is UMMF.*

E. Implementation

An advantage of such dynamics is their inherent distributed nature. They can be easily implemented distributively and asynchronously. There has been a considerable amount of work on distributive implementations of Waterfill in the single path setting (see e.g. [1]), and the standard of ABR traffic in ATM networks was designed to support it. The main difference here is that each router B has to know the flow of a commodity i along *all* its paths and not only along the path p that goes through B, in order to decide if i can increase along p.

Here is a high level sketch of a conceptual framework for such an implementation. Commodities would periodically send control packets (much like the RM cells in ATM) to the routers, which include their total flow value. Given this information, each router B can inform each commodity whether it can increase its rate (namely, if a commodity does not have the maximum rate among all the commodities sharing the router B, it can potentially increase the rate). A commodity i which has a path p on which all routers report that i can increase its rate, increases the rate along path p. Commodities can decrease their rates either by getting explicit messages from the routers or implicitly by periodically checking how much they can send along each path.

IV. Iterative Exhaustive Waterfill (IEWF)

In this section we give a natural generalization of the Waterfill algorithm and prove that it converges to an UMMF multicommodity flow. We call this generalization the Iterative Exhaustive WaterFill (IEWF) algorithm.

We start with an intuitive description of the algorithm, a precise description is provided in Section IV-A. The algorithm works in iterations. Each commodity maintains a distribution $\{\lambda(p) \mid p \in P_i\}$ over its paths p. We call the fractions $\{\lambda(p) \mid p \in P_i\}$, the *splits* of commodity i. These distributions are updated in each iteration. During an iteration we increase the flow of the commodities at the same rate and split it to paths according to λ. When edges (and thereby paths) gets saturated we re-scale the splits of the paths remaining open. The flow at the end of the iteration determines the splits with which we start the next one. The goal is to have splits for each commodity, such that all of its paths, with non-zero flow, will be saturated at the same time t, and all its paths with zero flow will be saturated by time t.

A. The algorithm

We now define the IEWF algorithm precisely. Given an instance of the multi-commodity flow problem IEWF starts with arbitrary initial splits $\lambda(p)$ for $p \in P_i$ such that $\sum_{p\in P_i} \lambda(p) = 1$ and $\lambda(p) \geq 0$. The IEWF updates the splits in each iteration as follows.

Its starts the iteration with all commodities sending 0 flow. Throughout the iteration it increases all commodities simultaneously at the same rate. To increase commodity i by 1, it increases the flow along paths $p \in P_i$ by $\lambda(p)$. When an edge e becomes saturated we set the splits of all paths p going through edge e to 0. When setting the split of a path $p \in P_i$ to zero there are two cases. If there is at least one non-saturated path with a non-zero split then we re-scale the splits of all other paths of commodity i proportionally to their previous value so they sum up to 1. Let UnSat be the set of unsaturated paths and Sat be the set of saturated paths. Then the new split of an unsaturated path p of commodity i is,

$$\lambda'(p) = \frac{\lambda(p)}{\sum_{p\in P_i \cap \text{UnSat}} \lambda(p)}$$

If all non-saturated paths have split zero then we change these splits to be equal and sum to one (i.e., each equals to one divided by the number of non-saturated paths).

An iteration terminates when all paths of all commodities are saturated. When an iteration terminates the IEWF algorithm uses the resulting flow to define the splits with which it starts the next iteration. That is, let $f_i(p)$ be the flow of commodity i along path $p \in P_i$ at the end of the iteration, then we set $\lambda(p) = \frac{f_i(p)}{v(f_i)}$ and start a new iteration.

The proof that the IEWF algorithm converges to a UMMF multicommodity flow is fairly complex and consists of the following steps.

1) We show that the flow and the splits maintained by IEWF have a limit (i.e., they converge to some values as we move from an iteration to the next). This step is described in Section IV-C.

2) We define a notion of Equilibrium Max-Min Fair (EMMF) multicommodity flow. This notion captures multicommodity flows F which define splits such that if we start an iteration of IEWF with these splits then the resulting flow is exacty F. We prove that a multicommodity flow is EMMF if and only if it is UMMF. This is described in Section IV-B.

3) We complete the proof by showing that the limit of IEWF is an EMMF multicommodity flow (and therefore UMMF). The proof of this last step is complicated since an iteration of IEWF, as a function mapping splits to splits is not a continuous function. This part is described in Section IV-D.

B. Equilibrium Max-Min Fairness (EMMF)

Given any multi-commodity flow F, and commodity i, we can define the splits $\lambda_i(p) = f_i(p)/v(f_i)$, $p \in P_i$ ($\sum_{p\in P_i} \lambda_i(p) = 1$) which are associated with commodity i in F. Suppose we run IEWF starting with these splits. For each edge e (resp. path p), let $\tau(e)$ (resp. $\tau(p)$) be the time in which e (resp. p) gets saturated during this run of IEWF. For an edge e let $flow(e,t)$ (resp. $flow(p,t)$) be the amount of flow through and edge e (resp. a path p) at time t during the run.

Definition IV.1 (EMMF)**.** *A feasible multi-commodity flow is* equilibrium max-min fair (EMMF) *if for every commodity i the followings hold:*

1) for every path $p \in P_i$, with $\lambda(p) > 0$ we get that $\tau(p) =$

$v(f_i)$.
2) for every p with $\lambda(p) = 0$, $\tau(p) \leq v(f_i)$.

Note that if we start with an EMMF flow F and run IEWF using the splits of F as the initial splits then for every commodity i all paths with nonzero flow are saturated exactly at $v(f_i)$. The paths of commodity i with 0 splits will be saturated by time $v(f_i)$. The following lemma follows from the definition of EMMF flow and from the definition of IEWF.

Lemma IV.2. *For each path $p \in P_i$ with $\lambda(p) > 0$ we have that $flow(p, v(f_i)) = \lambda(p)v(f_i)$. For each path p with $\lambda(p) = 0$ we have that $flow(p,t) = 0$ for any t.*

The following theorem shows that the notions of UMMF and EMMF are equivalent.

Theorem IV.3. *A flow is upward max-min fair if and only if it is equilibrium max-min fair.*

Proof: Assume that the flow F is equilibrium max-min fair. Since F is equilibrium max-min fair we know that for every commodity i and every path $p \in P_i$, there exists an edge e such that $\tau(e) \leq v(f_i)$. Recall that $L_i = \{f_j \mid v(f_j) > v(f_i)\}$. We claim that for any commodity $f_j \in L_i$ there is no path $p \in P_j$ with $\lambda_j(p) > 0$ that goes through e. Indeed, if there is such p we get that $\min_{e' \in p} \tau(e') \leq \tau(e) \leq v(f_i) < v(f_j)$ in contradiction with the assumption that F is equilibrium max-min fair (with respect to f_j).

This implies that e remains saturated even if we remove all flows f_j with $v(f_j) > v(f_i)$ from the graph. It follows that we cannot increase the flow on any path of commodity i even if we remove all flows f_j with $v(f_j) > v(f_i)$ from there graph so F is upward max-min fair.

For the converse, assume that the flow F is upward max-min fair. Then by Claim II.2, for every commodity f_i and path $p \in P_i$ there is an edge $e \in p$ such that

$$\sum_{p' \mid (e \in p') \wedge (p' \in P_j) \wedge (v(f_j) \leq v(f_i))} f_j(p') = c(e) \ .$$

Thus by time $v(f_i)$ this edge is saturated, or in other words $\tau(e) \leq v(f_i)$.

It is left to show that for every path p, if $\lambda_i(p) > 0$ then $\tau(p) = \min_{e \in p} \tau(e) = v(f_i)$. We prove this by contradiction. Let p' the path with minimum $\tau(p')$ among all paths $\{p \mid p \in P_j \text{ and } \tau(p) < v(f_j)\}$. Assume $p' \in P_i$. Then by the definition of IEWF until time $\tau(p')$, IEWF has not changed any splits. Let e be an edge on p' such that $\tau(e) = \tau(p')$. Then $flow(e, \tau(p)) = \sum_{p \mid e \in p} \lambda(p)\tau(p) < \sum_{p \mid e \in p, p \in P_j} \lambda(p)v(f_j) \leq c(e)$. The strict inequality holds since $\tau(p') < v(f_i)$ and $e \in p'$, and the last inequality holds since F is feasible. This contradicts the assumption that e is saturated at time $\tau(p')$. ∎

C. Convergence of IEWF

In this section we show that the multi-commodity flow computed in in each iteration of IEWF converges to a limit. In the next section we show that this limit is an EMMF multicommodity flow (and hence UMMF).

Consider an iteration of IEWF. Recall that $flow(p,t)$ is the flow on path p at time t. Let g_i^0 be the flow of commodity i at the end of the previous iteration whose splits are used to start the current iteration. We define a flow function g_i^t by setting $g_i^t(p) = flow(p,t)$ if $p \in P_i$ is saturated at time t and $g_i^t(p) = g^0(p)$ if $p \in P_i$ is not saturated at time t. Let $\lambda(p,t)$ be the split that IEWF uses for a path p at time t. We define $v(g_i^t) = \sum_{p \in P_i} g_i^t(p)$. Note that $v(g_i^0)$ is the total flow of commodity i in the previous iteration. Since we are discussing an arbitrary commodity i in

Lemma IV.4. *Fix an unsaturated path p of commodity i and some time t. Then $g_i^0(p) \geq flow(p,t)$ if and only if $v(g_i^t) \geq t$. Furthermore, if $g_i^0(p) \geq flow(p,t)$ then*

$$g_i^0(p) - flow(p,t) = \lambda(p,t)(v(g_i^t) - t).$$

Due to lack of space we omit the proof.
Consider an iteration of IEWF. Let $t_1, t_2, \ldots, t_{m'}$ be the times in which edges get saturated during this iteration. Let E_i be the set of edges that are saturated at time t_i and let G^{t_i} be the multi-commodity flow after the saturation of E_i. Let G^{t_0} be the multi-commodity flow at the beginning of the iteration.

We consider the sequence $G^{t_0}, G^{t_1}, \ldots, G^{t_{m'}}$. The difference between G^{t_i} and $G^{t_{i+1}}$ is in the value of the flow along all paths that go through an edge of E_{i+1} and were not saturated before time t_{i+1}. In $G^{t_{i+1}}$ we change the flow value along each such path to be the value that actually flows along it in the flow maintained by IEWF (instead of its value in G^{t_i}).

Consider the sequence obtained from $G^{t_0}, G^{t_1}, \ldots, G^{t_{m'}}$ by splitting the transition from G^{t_i} to $G^{t_{i+1}}$ into multiple steps each changing the value along a single path that gets saturated at t_{i+1}. We first in increase the flow along paths whose flow increases in an arbitrary order and then decrease the flow along paths whose flow decreases in an arbitrary order. We show that this refinement is a *happy* sequence. To simplify the following presentation, we ignore the refinement of each transition from G^{t_i} to $G^{t_{i+1}}$ to many transitions each changing a single path. We also abuse the notation slightly and refer to $G^{t_0}, G^{t_1}, \ldots, G^{t_{m'}}$ as a *happy* sequence (meaning that its refinement is happy).

It then follows that the concatenation of the sequences $G^{t_0}, G^{t_1}, \ldots, G^{t_{m'}}$ of all the iterations of IEWF is also a *happy* sequence and therefore by Corollary III.2 the flow maintained by IEWF has a limit. Furthermore, since G^{t_0} is a feasible flow it follows by Corollary IV-C that the limit of IEWF is feasible.

Theorem IV.5. *For each iteration of IEWF the sequence $G^{t_0}, G^{t_1}, \ldots, G^{t_{m'}}$ is* happy.

Proof: Each flow G^{t_i} is feasible when we multiply the capacities by a factor of 2. This follows since it is a sum of a flow contained in G^{t_0} which is feasible, and a subset of the flow maintained by IEWF which is also feasible. This establishes that there is an $R > 0$ that bounds the flow of each commodity, and establishes the first property of a happy sequence. The second property, of a single path modification

is immediate from the construction. It remain to construct the mapping Π (property 3) and relate the increases to the decreases (property 4).

Consider the transition from G^{t_i} to $G^{t_{i+1}}$ and an edge $e_{i+1} \in E_{i+1}$. Let P^- be the set of paths through e_{i+1} that are being saturated at t_{i+1} such that the flow through them in $G^{t_{i+1}}$ is smaller than the flow through them in G^{t_i}. Let P^+ be the set of paths through e_{i+1} saturated at t_{i+1} *or before* such that the flow through them in $G^{t_{i+1}}$ is larger than the flow through them in G^{t_0}.

Let $\Delta^- = \sum_{p \in P^-} decrease(p)$, where $decrease(p)$ is the amount by which the flow along p decreases. Let $\Delta^+ = \sum_{p \in P^+} increase(p)$, where $increase(p)$ is the amount by which the flow along p increases. We claim that $\Delta^+ \geq \Delta^-$. This holds since $\Delta^+ - \Delta^-$ is larger than the difference between the flow along e_{i+1} in $G^{t_{i+1}}$ and the flow along e_{i+1} in G^{t_0} which is nonnegative since the edge e_{i+1} is saturated in $G^{t_{i+1}}$.

Since $\Delta^+ \geq \Delta^-$ we can map each unit of decrease in the value of a path in P^- to a unit of increase in the value of a path in P^+. When we go over all edges $e_{i+1} \in E_{i+1}$ this mapping defines the function Π. Note that a path p can be counted as an increase in multiple edges $e_{i+1} \in E_{i+1}$, but clearly at most $m = |E|$ edges. Therefore, every unit of increase in the flow value along a path p is being charged by at most m unit of decrease.

Assume that at we charge a decrease in the flow along a path p of commodity j to an increase in the flow along a path p' of commodity j'. Let $t' \leq t_{i+1}$ be the time in which p' got saturated. (It follows that at time t' we increased the flow along p'.) Let r be the rank of commodity j at time t_{i+1}, and let r' be the rank of commodity j' at time t'. It remains to show that $r' < r$.

By Lemma IV.4, $v(g_j^{t_{i+1}}) \geq t_{i+1}$. It follow that also for every commodity β of rank larger than r at time t_{i+1} we have $v(g_\beta^{t_{i+1}}) \geq t_{i+1}$. Furthermore from Lemma IV.4 we can also deduce that the value of all commodities of rank at least r at time t_{i+1} was at least t_{i+1} since the beginning of the iteration.

By Lemma IV.4, $v(g_{j'}^{t'}) < t'$. Since $t' \leq t_{i+1}$ we get that all flows of rank at least r at time t_{i+1} are larger than j' at time t' and therefore $r' < r$. ∎

D. The limit of IEWF is equilibrium

If we think of an iteration of IEWF as a function f mapping splits to splits then each iteration of IEWF computes $\lambda \leftarrow f(\lambda)$. It would be easy to prove that an iterative algorithm of this sort (which we already know that is converging) indeed converges to a fixed point of f (EMMF) if f is continuous. Unfortunately f is not continuous. There are examples where a tiny perturbation of initial conditions can reach a very different outcome in IEWF.

Despite this discontinuity we prove that

Theorem IV.6. *The limit of the sequence of flows produced by the iterations of IEWF is an EMMF multicommodity flow.*

The proof of this theorem is by contradiction. We show that if the limit (which we already know that exists) is not EMMF then when the flow of IEWF is already very close to it, the change in the splits is too large. The details are delicate and we omit them due to lack of space.

V. Experimental Study

We analyzed the behavior of IEWF on two types of networks: the Google backbone network and a set of synthetic networks generated using the Waxman model [11] in varying sizes. For the Google backbone experiments, we used a metro level abstraction of Google's backbone. We selected a subset of the largest demands in the Google network as our commodity set.

For the Waxman graphs we generated 20 graphs with 20, 30, 40, 50, 60 and 70 nodes. We used $\alpha = 0.55$ and $\beta = 0.55$, following the guidelines from [12]. These parameters gives us graphs with an average degree of about $0.3n$, where n is the number of nodes. For each size we generated graphs repetitively, discarding graphs which are not two connected, until we got 20 two connected graphs.

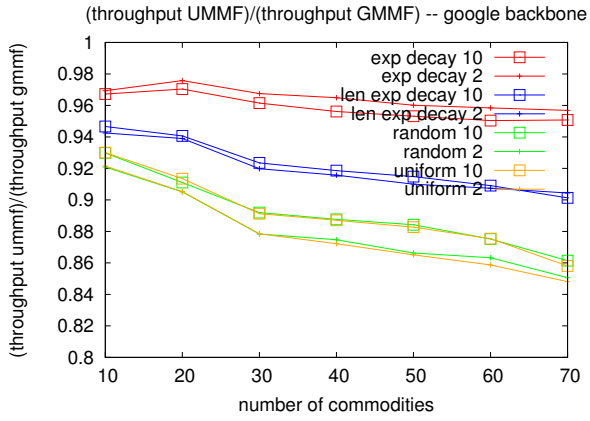
To generate the commodities for the Waxman graphs we randomly split the nodes into three sets and selected one as the sources and the other as the destinations. We defined a commodity from each of the sources to each of the destinations. Overall, there were about $(\frac{n}{3})^2$ commodities for each graph.

As explained in the introduction, the global max-min allocation (GMMF) can be computed by iterative deployment of a linear program. We implemented this algorithm and used the flows it produces in order to evaluate the behavior of IEWF.

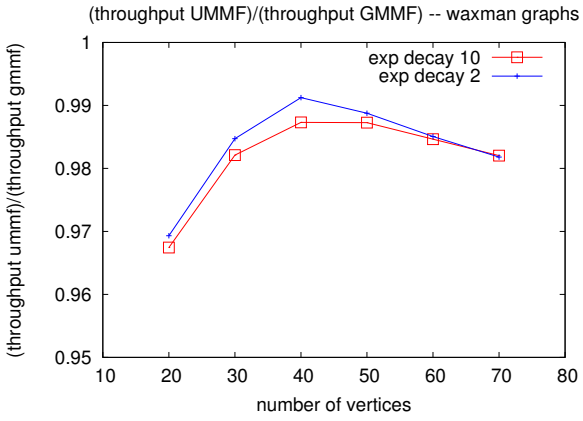
A. IEWF Split Selection and Convergence

In our first experiment we study the behavior of IEWF w.r.t. initial split selection and the convergence rate in practical scenarios. We compare four methods of split initialization for the IEWF: The first two methods are oblivious to the path characteristics; in the uniform method splits are defined simply as $\frac{1}{\text{\# of paths for the commodity}}$, and in the random method, splits are computed proportionally to a set of random numbers. The other two methods use decaying splits, both routing more flow along shorter paths. The first variant, called *len exponential decay* uses splits proportional to $1/10^\ell$, where ℓ is the number of hops in the path. This method prefer paths with less hops by giving them a greater split, but all paths with the same hop count get the same split. The second variant adds a tie breaking component to paths with equal hop count as follows; sort the paths for each commodity from shortest to longest by hop count. If two or more paths have the same number of hops, arbitrarily order them. Let i be the index of a path p for some commodity. The split given to this path is proportional to $1/10^i$. The result here is an exponential decay in the split values where no two paths belonging to the same commodity have the same split. We call this split selection *exponential decay*.

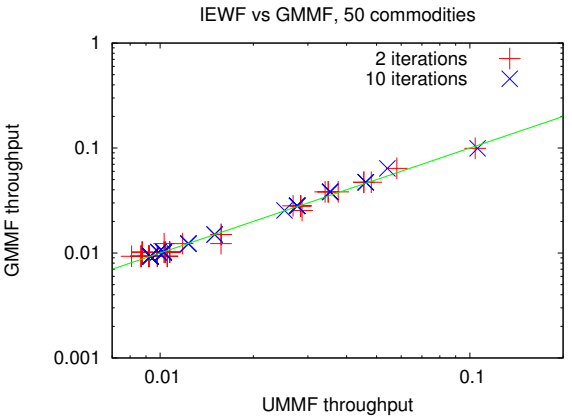
Figure 2(a) depicts the ratio between the overall throughput obtained by the IEWF algorithm after two and ten iterations, and the total throughput achieved by GMMF for the same



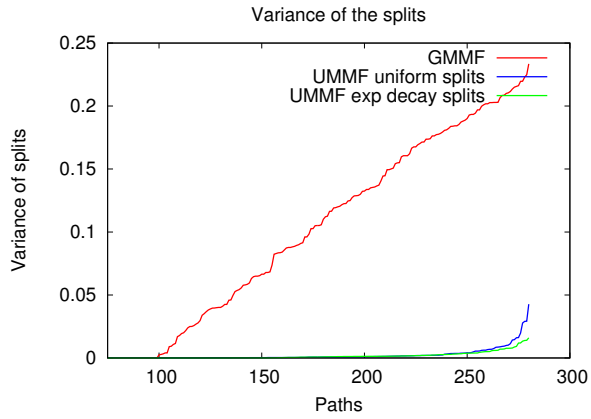
(a) Total throughput for the various initial conditions.



(b) Total throughput for the various initial conditions.



(c) Comparison of the flow values allocated by GMMF and IEWF (with 2 and 10 iterations).



(d) Comparison of the path allocation variance.

Fig. 2. Experimental Results

input. Each point in the graph is the average of 50 runs, each having a random subset of 50 commodities. The x-axis represents the number of commodities in this random subset, and the y-axis represents the average total throughput ratio.

It is clear from the graph that the approaches that take into account the paths' length perform better in terms of higher overall throughput. The reason is that less links are used when shorter paths are preferred and the network, in general, is less congested, hence more traffic can be routed (assuming there are many commodities in the network). The highest throughput was achieved when using exponential decay. The reason is that this approach prefers exactly one path per commodity over the others and will route significant amounts of traffic on each path only when all the shorter paths for this commodity are fully congested. Again, the network is less congested in this scenario as paths get saturated one by one, leaving larger parts of the network "free".

The main difference between runs of 2 and 10 iteration (which is not reflected by the total throughput) is in the fairness between commodities. A run of 10 iteration is able to better balance the "equivalent" commodities (commodities that have the same flow in GMMF) in the following sense. Commodities that have the same flow value in a GMMF solution typically have closer values after 10 iterations than after 2 iterations. The gain in the throughput of the run of 2 iterations, compared to the run of 10 iterations, with exponential decay splits, is mainly due to very few high flow commodities .

We also conducted this experiment on the family of Waxman graphs. The results are shown in Figure 2(b). The x-axis represents the number of nodes in the graph and the y-axis is the average of the UMMF throughput divided by the GMMF throughput on all the graphs tested. We used exponential decay as the split selection and compared the results after two and ten iterations. On these graphs, the UMMF solution reached at least 96% of the GMMF solution's throughput.

Similar to the Google backbone graph, one can see the expected slow decline in the quality of the UMMF solution, as the network grows larger. In this case, we also see an improvement, going from 20 to 50 nodes. This improvement is due to the fact that we required the Waxmann graph to be 2-connected. For small graphs, this means that we had to reject many of the graphs we sampled, and this skews the distribution.

B. Allocations of Individual Commodities

Figure 2(c) depicts a typical run over a random set of 50 commodities in the Google backbone network. Each point in the graph represents one commodity out of the top Google demands. The x coordinate represents the ratio between the throughput given to that commodity in the IEWF solution we found (either 2 or 10 iterations) to the total throughput in the GMMF solution. The y coordinate represents the ratio between the throughput given to the same commodity in the GMMF solution found by the LP based algorithm to the total throughput in the GMMF solution. The points along the diagonal represent commodities that got the same throughput both in the GMMF solution and in the IEWF solution, points below the diagonal represent commodities with a bigger share in the IEWF solution, while points above the diagonal represent commodities with a bigger share in the GMMF solution.

As can be clearly seen from Figure 2(c) in the practical scenarios we considered, the results of GMMF and UMMF were close, and there is no significant gap in fairness. Running 10 iterations usually outperforms 2 iterations in terms of fairness. Going from 10 to 50 iterations did not produce much of a difference, and thus is not depicted here.

C. IEWF Stability

Finally, we tested the stability of the two approaches on the Google backbone network to see how sensitive is the resulting multicommodity flow to a small change in the demands. In real life, demands vary over time (usually in a rather smooth manner), which leads to changes in the Traffic Engineering solution deployed in the network. An important property of any TE algorithm is stability when reacting to these changes, as deploying new splits or paths to the network is time consuming and can have undesirable effects like out of order arrival of TCP packets causing re-transmits.

For this experiment, we used the Google backbone network and chose 50 different sets of demands, which are very similar. For each set, we found the GMMF solution, and the IEWF solution (both with random initial splits and with exponentially decaying splits). For each commodity and path, we collected the splits of this path from all the experiments in which the commodity participates, and computed its variance. The results are presented in Figure 2(d): the top line in the graph represents a histogram of the split variance in the GMMF solution and the other two lines represent a histogram of the variance values in the IEWF runs. The variance in the splits generated by IEWF was significantly lower than the splits variance in the GMMF solution. One can conclude that a small change in the demand set led to a rather big change in the GMMF solution, while the IEWF solution was more stable.

VI. Related Work

Ever since the fundamental work of Jain et. al. [7] originally published in 1984, the notion of fairness was the subject of extensive research in the context of routing, flow control, and more recently traffic engineering [3], [13], [8].

In this paper we concentrate on max-min fairness in the context of multicommodity flow. A recent survey focusing on this topic can be found in [8], indicating that all advanced algorithmic solutions require, in some way or another, to solve linear programs iteratively.

Two papers are of prime interest. In [9] the authors define a generalized notion of bottleneck, and show that a generalization of the Waterfill algorithm can be applied in the case of a multicommodity flow. However, this generalization requires an iterative solution of linear programs.

Another recent approach to circumvent the computational difficulty of the problem by using a relaxed notion of max-min fairness was studied in [2]. The authors define what they call a "local" max-min fair flow, and present a fast algorithm that computes a flow which approximates this local max-min fair flow. Their approach relies on recent approximation algorithms for concurrent multicommodity flow [6], [5]. Unfortunately, their definition of a local max-min fair multicommodity flow is weak and not natural, since it depends on the algorithm which is used to compute concurrent multicommodity flow. As a result the exact properties of the approximate allocations are not clear.

Our algorithmic approach is based on a generalization of the simple distributed single path scheme, which naturally leads to the definition of upward max-min fair multicommodity flow. Moreover, the definition of UMMF is independent of the algorithm used, and we actually prove that a large family of dynamics converge to it.

References

[1] Y. Afek, Y. Mansour, and Z. Ostfeld. On the convergence of rate based flow control. In *STOC*, pages 106–143, 1996.

[2] M. Allalouf and Y. Shavitt. Centralized and distributed algorithms for routing and weighted max-min fair bandwidth allocation. *IEEE/ACM Transactions on Networking*, 16(5):1015 –1024, oct. 2008.

[3] D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, Englewood Cliffs, 2001.

[4] Y. Dinitz. Dinitz' algorithm: The original version and Even's version. In *Essays in Memory of Shimon Even*, pages 218–240, 2006.

[5] L. K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13:505–520, 2000.

[6] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2):630–652, 2007.

[7] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Computing Research Repository*, cs.NI/9809, 1998.

[8] D. Nace and Pioro M. Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial. *IEEE Communications Surveys and Tutorials*, 10(4):5 –17, 2008.

[9] B. Radunovic and J. Le Boudec. A unified framework for max-min and min-max fairness with applications. In *40th Annual Allerton Conference on Communication, Control, and Computing*, 2002.

[10] N. Wang, K. Ho, G. Pavlou, and M. Howarth. An overview of routing optimization for internet traffic engineering. *Communications Surveys Tutorials, IEEE*, 10(1):36 –56, quarter 2008.

[11] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communication*, 6(9):1617–1622, 1988.

[12] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *INFOCOM*, pages 594–602, 1996.

[13] Y. Zhou. *Resource Allocation in Computer Networks: Fundamental Principles and Practical Strategies*. PhD thesis, Drexel University, 2003.