Upward Max-Min Fairness

EMILIE DANNA and AVINATAN HASSIDIM, Google
HAIM KAPLAN, Tel Aviv University
ALOK KUMAR, Google
YISHAY MANSOUR, Tel Aviv University and Microsoft Research
DANNY RAZ, Technion
MICHAL SEGALOV, Google

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 Multiprotocol Label Switching (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, and is hard to implement in a distributed environment. More importantly, a network operator has limited control and understanding of the inner working of the algorithm.

In this article 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.

Categories and Subject Descriptors: C.2.4 [Computer-Communication Networks]: Distributed Systems; F.2.2 [Analysis of algorithms and problem complexity]: Nonnumerical Algorithms and Problems

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Max min fairness, waterfill, multicommodity flow upward max min fair multicommodity flow, iterative exhaustive waterfill

A preliminary version was presented in INFOCOM 2012 and received the best article runner-up award.

This work was partially done when Yishay Mansour, Haim Kaplan, and Danny Raz were at Google.

H. Kaplan's research was partially supported by the Israel Science Foundation grants 822-10 and 1841-14, the German-Israeli Foundation for Scientific Research and Development (GIF) grant no. 1161/2011, and the Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11). Y. Mansour was supported in part by a grant from the Israel Science Foundation, a grant from the United States-Israel Binational Science Foundation (BSF), a grant by Israel Ministry of Science and Technology and the Israeli Centers of Research Excellence (I-CORE) program (Center No. 4/11).

Authors' addresses: E. Danna; email: edanna@google.com; A. Hassidim; email: avinatan@google.com; H. Kaplan; email: haimk@cs.tau.ac.il; A. Kumar; email: kumarak@google.com; Y. Mansour; email: mansour@tau.ac.il; D. Raz; email: danny@cs.technion.ac.il; M. Segalov; email: msegalov@google.com.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.

© 2017 ACM 0004-5411/2017/03-ART2 \$15.00

DOI: <http://dx.doi.org/10.1145/3011282>

ACM Reference Format:

Emilie Danna, Avinatan Hassidim, Haim Kaplan, Alok Kumar, Yishay Mansour, Danny Raz, and Michal Segalov. 2017. Upward max-min fairness. *J. ACM* 64, 1, Article 2 (March 2017), 24 pages.
DOI: <http://dx.doi.org/10.1145/3011282>

1. 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 Transmission Control Protocol (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) [Bertsekas and Gallager 2001]. 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 Internet Protocol (IP), see for example, Wang et al. [2008]. When we start, all commodities are active. We increase the flow of all commodities equally until the first link (or links) in the network is 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) [Bertsekas and Gallager 2001]. 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 who gets an equal or less amount. In other words, a multicommodity flow is max-min fair if the sorted vector of flow values (one number per commodity) is the lexicographically largest feasible sorted vector of flow values. This vector is known to be unique [Bertsekas and Gallager 2001].

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, then the dynamic performs the change. This simple dynamic is guaranteed to converge to the max-min fair solution [Afek et al. 1996].

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

When traffic can be sent along multiple paths between the source and the destination, the situation becomes much more complex. This is the case in many practical traffic engineering scenarios, where Internet Service Provider (ISP) are using multiple paths (for example, using Multiprotocol Label Switching (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 of the commodities 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 corresponds to this vector.

One can extend the distributed dynamic that 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 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 article, 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 what we call an *Upward Max-Min Fair (UMMF)* multicommodity flow. This new fairness notion has many intuitive fairness guarantees.

1.1. Our Contribution

1.1.1. UMMF—A New Notion of Fairness. We define a multicommodity flow to be UMMF, if, for each i , we cannot increase the value of the i th smallest commodity, along any of its paths, even if we remove all commodities whose value is strictly larger than the i th smallest value. (This implies that if we reach a UMMF flow, then the dynamic described earlier terminates.) This is a new fairness concept that fundamentally differs from previous ones. It can be achieved by simple and natural algorithms with an elegant mathematical analysis.

At 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 a UMMF flow, the requirement is that we cannot increase the i th smallest commodity while fixing the routing of the flow 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 value of the i th smallest commodity in 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 to the i th smallest commodity so the i th smallest commodity increases, then the flow would not be max-min fair but it may be upward max-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 a multicommodity flow satisfying the traditional stronger notion as a *Global Max-Min Fair (GMMF)* multicommodity flow.

The example in Figure 1(a) highlights the differences between UMMF and GMMF flows. In this example, the only global max-min fair flow is the one with (1, 1) as the

Fig. 1. (a) Two commodities, each with two paths, $P_1 = \{(s_1ABt_1), (s_1ABCt_1)\}$ and $P_2 = \{(s_2, BCt_2), (s_2ABCt_2)\}$, capacities are 1. (b) A network with $2n + 2$ vertices. One commodity with $n + 1$ paths from s to t : n paths of length three each using a different edge of the middle layer, and one long path (dashed) of length $2n + 1$. Capacities are 1. The blocking flow that sends one unit of flow along the long path is a factor of n smaller than the maximum flow.

vector of sorted flow values. On the other hand, there are many different UMMF flows with different vectors of sorted flow 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.

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 Dinitz’s maximum flow algorithm [Dinitz 2006]).

The example in Figure 1(a) can be extended to show that, in the worst case, a UMMF flow can have a total throughput that is a factor of $\Theta(n)$ smaller from the total throughput of a GMMF flow, where n is the number of nodes. In fact, this may be the gap in the worst case, even when we have a single commodity, as we see in Figure 1(b). We see a blocking flow that routes only one unit of flow on a single long path, where the maximum flow is of size $\Theta(n)$ and routes on the short “parallel” paths of length 3. For a single commodity, we can prove that the gap between the value of a maximal flow and the value of a maximum flow cannot be larger than $O(n)$. To see this, decompose the maximum flow M and the blocking flow B into simple paths. For each edge e , saturated by B , we charge the flow through paths of M containing e to the paths of B containing e such that no path of B is charged more than the amount of flow which it carries. By the definition of a blocking flow, each path of M must contain an edge saturated by B , so we have charged all the flow along the paths of M to the paths of B . Since each path of B contains at most $n - 1$ edges, it has been charged at most n times.

Although the possible large worst-case gap in terms of total throughput between GMMF and UMMF multicommodity flow, we find the UMMF concept very interesting. For one, the gaps in the total throughput between GMMF and UMMF multicommodity flow in our experiments were very small (typically $< 10\%$), and as two fairness concepts

²A blocking flow is a flow that we cannot increase without rerouting some part of it.

their main target is fairness and not throughput. Furthermore, computing a UMMF multicommodity flow can be achieved by simple algorithms with elegant analysis as we discuss next.

1.1.2. Distributed Dynamics. We give a set of simple conditions such that any dynamic obeying these conditions converges 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.

1.1.3. 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.

We suggest a natural extension of the Waterfill algorithm, which we call *Iterative Exhaustive WaterFill (IEWF)*, for the case of multiple paths per commodity.

An iteration of our IEWF algorithm is called *Exhaustive Waterfill*. Exhaustive Waterfill takes as input *splits* for each commodity that indicate how to split an increase of the commodity along its paths. These splits are fractions, one per path of the commodity, that sum to 1. We start increasing all commodities at the same rate while using the splits to distribute the increase along the paths of each commodity. When an edge e is saturated, then we take the paths containing e out of the game. We adjust the splits of the remaining paths of each commodity so they are proportional to the original splits and sum to 1. We then continue increasing the commodities that still have open paths³ until another edge gets saturated, and so on. The algorithm terminates when all paths go through saturated edges.

Our IEWF algorithm starts with some initial splits (say, uniform splits) and runs Exhaustive Waterfill using these splits. Then it takes the resulting flow and computes for each commodity the fraction of the flow that it sends along each of its paths. The fractions are used as the input splits for another iteration of Exhaustive Waterfill. We stop when we either reach splits that are a fixed point of Exhaustive Waterfill or after a certain number of iterations.

We prove that IEWF converges to a UMMF multicommodity flow. This, in fact, reinforces our claim that the notion of UMMF multicommodity flow is a natural one.

Another computational advantage of a UMMF multicommodity flow over the traditional GMMF multicommodity flow is that the algorithms that we provide to compute (or rather approximate) it are combinatorial and do not need to solve linear programs. In contrast, there is no combinatorial algorithm to find (or approximate) a GMMF multicommodity flow when there are multiple possible paths per commodity. Finding such an algorithm is a challenging open problem. Still, it is possible to find a GMMF flow by solving multiple linear programs, at least one to discover every flow value [Nace and Pioro 2008]. Although this algorithm 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 a limited control and understanding of the inner working of the algorithm. This approach is hard to distribute, and thereby 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.

1.1.4. Experimental Study. We performed an experimental study to check various properties of UMMF multicommodity flows and the IEWF algorithm: We conducted our

³An open path is a path that does not contain a saturated edge.

tests on the Google backbone network [Jain et al. 2013] and on random networks drawn using Waxman's model [Waxman 1988; Zegura et al. 1996]. 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 that we obtain (even after 2 iterations and definitely after 10) is close to the quality of the GMMF multicommodity 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 identify the one that gives the best performance. We also compared how sensitive are the GMMF and the UMMF multicommodity flows to small changes of their input. Our results show that the UMMF flow is much more robust.

1.2. Related Work

Ever since the fundamental work of Jain et al. [1998] originally published in 1984, the notion of fairness has been the subject of extensive research in the context of routing, flow control, and, more recently, traffic engineering [Bertsekas and Gallager 2001; Zhou 2003; Nace and Pioro 2008].

In this article, we concentrate on max-min fairness in the context of multicommodity flow. A recent survey [Nace and Pioro 2008] on this topic indicates that all advanced algorithmic solutions require, in some way or another, to solve linear programs iteratively.

Two articles are of prime interest. Radunovic and Boudec [2002] 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 Allalouf and Shavitt [2008]. Allalouf and Shavitt [2008] 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 [Garg and Könemann 2007; Fleischer 2000]. Unfortunately, their definition of a local max-min fair multicommodity flow is weak and not natural, since it depends on the algorithm that 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 flow is independent of the algorithm used, and we actually prove that a large family of dynamics converge to it.

2. 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 that commodity i can use. (Note that P_i can potentially contain 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 $(f_1, f_2, \dots, f_k)$ of flows f_i , $1 \leq i \leq k$. The multicommodity flow is *feasible* if for every edge $e \in E$, we have $\sum_i \sum_{p \in P_i | e \in p} f_i(p) \leq c(e)$.

Consider a multicommodity flow $H = (h_1, \dots, h_k)$. Let $\sigma(i)$ be the flow with the i th 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 $v(H) = (v(h_{\sigma(1)}), v(h_{\sigma(2)}), \dots, v(h_{\sigma(k)}))$ be 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.

Definition 2.1 (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, whose proof is straightforward and hence omitted, gives an alternative formulation of a UMMF flow.

CLAIM 2.2. *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*

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

3. THE LEXICOGRAPHIC-INCREASE DYNAMICS

3.1. 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 of higher flow value. We require that the value of the increased commodity (after the increase) will not exceed the value of each decreased commodity (after the decrease). We prove that any such a dynamic converges to a limit and that any limit of such a dynamic 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, and the main challenge is to bound the number of decrease operations.

The proof focuses on the sorted vector of flow values $v(H) = (v(h_{\sigma(1)}), \dots, v(h_{\sigma(k)}))$ of a multicommodity flow H , defined in Section 2. Note that the identity of $\sigma(i)$, the i th smallest 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, by the definition of the dynamic, the smallest flow value never decreases. Now consider the second smallest flow value. This value might decrease in certain time steps. However, by the definition of the dynamic, 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 flows to increases of flow values of smaller flows.

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 dynamic 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 that is feasible, and 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 $\sigma^{-1}(j)$).

3.2. Definitions

Consider a sequence $H^1, H^2, \dots$ 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 every i we have $v(h_{\sigma_s(i)}^s) \leq v(h_{\sigma_s(i+1)}^s)$ and $v(h_{\sigma_s(i)}^{s+1}) \leq v(h_{\sigma_s(i+1)}^{s+1})$.
- (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.

3.3. Happy Sequence Converges

THEOREM 3.1. *If H^s is a happy sequence, then $v(H^s)$ has a limit.*

PROOF. Recall that $\sigma(i)$ is the commodity with the i th smallest value. Consider $v(h_{\sigma(1)}^s)$. 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 $v(h_{\sigma(1)}^s)$ does not decrease.

We show that the sum of the decreases of $v(h_{\sigma(2)}^s)$ is bounded: Since the total increase of $v(h_{\sigma(1)}^s)$ is at most R , the sum of the decreases of $v(h_{\sigma(2)}^s)$ is at most mR . Furthermore, since $v(h_{\sigma(2)}^s)$ is also bounded by R , the sum of the increases of $v(h_{\sigma(2)}^s)$ is at most $R + mR$. It follows that both the sum of increases and the sum of the decreases of $v(h_{\sigma(2)}^s)$ are bounded, and therefore each of these sums has a limit. This implies that $v(h_{\sigma(2)}^s)$ has a limit that is equal to the difference between the limit of the sum of its increases and the limit of the sum of its decreases.

In general, let A_i be an upper bound on the sum of the increases of $v(h_{\sigma(i)}^s)$. 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 $v(h_{\sigma(i)}^s)$ are bounded, and therefore they have a limit and $v(h_{\sigma(i)}^s)$ has a limit. $\square$

Theorem 3.1 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 giving the flow values along each path of each commodity 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.

COROLLARY 3.2. *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 of feasible flows, then H^* is a feasible flow.*

PROOF. From the proof of Theorem 3.1 follows that the sum of the increases of $v(h_{\sigma(i)}^s)$ is bounded by A_i and the sum of the decreases of $v(h_{\sigma(i)}^s)$ is bounded by, say, B_i . Let $A = \sum_{i=1}^k A_i$, and let $B = \sum_{i=1}^k B_i$. Let p be a path of some fixed commodity. Since at each step H^s changes only on a single path of a single commodity it follows that the sum of the increases in the amount of flow sent along p is bounded by A and the sum of the decreases in the amount of flow sent along p is bounded by B . This implies that each coordinate of $p(H^s)$ has a limit.

Any infinite subsequence of H^s must also converge to H^* , in particular the infinite subsequence of feasible flows converges to H^* . Since the set of feasible flows is compact it follows that H^* must be a feasible flow. $\square$

3.4. From a Happy Sequence to Dynamic

Consider a dynamic in which, as long as we have a flow H that is not UMMF, we pick some commodity i with flow h_i and increase its value along some of its paths while decreasing the flow along paths of commodities j with flow h_j such that $v(h_j) > v(h_i)$. Let h'_i denote the flow of commodity i after the increase and let h'_j denote the flow of 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 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 dynamic. 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 3.2, 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 derives the convergence.

THEOREM 3.3. *For any dynamic satisfying Conditions (1)–(5) above, the sequence of multicommodity flow H^s converges to a feasible multicommodity flow H^* that is UMMF.*

PROOF. As we mentioned, from Conditions (1)–(3) it follows that a refinement of the sequence is a happy sequence, and therefore the sequence H^s converges to a limit H^* .

Assume that H^* is not UMMF. Then there is a commodity i and a set of commodities J such that we can increase h_i along some path by at least ϵ while only decreasing flows h_j for $j \in J$, such that for every $j \in J$, we have $v(h_j^*) > v(h_i^*)$. It is easy to verify that we can scale ϵ down such that $v(h_j^*) - v(h_i^*) > 2\epsilon$.

Let $\delta \ll \epsilon$, which we will specify more precisely below, and let T_δ satisfy that for any $s > T_\delta$, H^s is δ close to its limit H^* . This means that the flow value along any path in

H^s differs by at most δ from its value at H^* . Such T_δ must exist, since H^* is the limit of the dynamic.

It follows that, for any $s > T_\delta$, we can increase h_i in H^s by at least $\epsilon - |P|\delta$ while decreasing flows h_j , such that $v(h_j) > v(h_i)$ to some value larger than the new value of h_i . ($P = \cup_i P_i$ is the set of paths of all commodities.) By Condition (4), there is a step $\tau > T_\delta$ such that the dynamics will increase flow h_i at step τ . By Condition (5), at step τ , our dynamic will increase h_i by at least $\zeta(\epsilon - |P|\delta)$. If we pick δ such that $\zeta(\epsilon - |P|\delta) \gg 2\delta$, then this contradicts the definition of T_δ . $\square$

3.5. Implementation

An advantage of such a dynamics is their 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, for example, Afek et al. [1996]), and the standard of Available Bit Rate (ABR) traffic in Asynchronous Transfer Mode (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 Resource Management (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 , then it can potentially increase the rate). A commodity i that 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.

4. ITERATIVE EXHAUSTIVE WATERFILL

In this section, we give a natural generalization of the Waterfill algorithm and prove that it converges to a UMMF multicommodity flow. We call this generalization the IEWF algorithm.

We start with an intuitive description of the algorithm, and a precise description is provided in Section 4.1. The algorithm works in iterations. Each commodity j maintains a distribution $\{\lambda(p) \mid p \in P_j\}$ over its paths p . We call the fractions $\{\lambda(p) \mid p \in P_j\}$, the *splits* of commodity j . 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 . To simplify the presentation, we introduce the notion of *time* and increase the flow of each commodity (that still have unsaturated paths) by one unit flow in each time unit.

When edges (and thereby paths) are saturated, we re-scale the splits of the paths that remain 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 .

4.1. The Algorithm

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

We start the iteration with all commodities sending 0 flow. Throughout the iteration, we increase all the commodities simultaneously at a rate of one unit of flow per time unit (time is set to 0 at the beginning of the iteration). To increase commodity j by one unit, we increase the flow along each path $p \in P_j$ by $\lambda(p)$.

When an edge e becomes saturated at some time t , we change the splits of the paths of the commodities that have paths through e as follows. Let j be a commodity with at least one path $p \in P_j$ that goes through e . We set the split of every path $p \in P_j$ that goes through e to 0. If not all paths in P_j are saturated at time t , then we continue according to one of the following cases:

Case 1: If commodity j has at least one unsaturated path with a non-zero split, then we re-scale the splits of all unsaturated paths of commodity j proportionally to their previous values so they sum 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 j is

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

Case 2: If all unsaturated paths of commodity j have split zero, then we change these splits to be equal and sum to 1 (i.e., each equals to one divided by the number of unsaturated paths).

Once we completed updating the splits, we continue to increase the flows (that still have unsaturated paths) at a rate of one unit of flow per time unit. (The time resumes moving forward from time t when the saturation of e occurred). Treating the case where more than one edge get saturated simultaneously is analogous.

An iteration terminates when all paths of all commodities are saturated. When an iteration terminates, the IEWF algorithm uses the resulting flow to *derive* the splits with which it starts the next iteration. That is, let $f_j(p)$ be the flow of commodity j along path $p \in P_j$ at the end of the iteration, and then we set $\lambda(p) = f_j(p)/v(f_j)$ and start a new iteration with these splits.

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

- (1) We define a notion of Equilibrium Max-Min Fair (EMMF) multicommodity flow. This notion captures multicommodity flows F that define splits such that if we start an iteration of IEWF with these splits, then the resulting flow is exactly F . We prove that a multicommodity flow is EMMF if and only if it is UMMF. This is described in Section 4.2.
- (2) 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 4.3.
- (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 not immediate, since an iteration of IEWF, as a function mapping splits to splits, is not a continuous function. This part is described in Section 4.4.

Remark. One peculiar aspect of IEWF is its special treatment of paths with 0 splits. It is not clear why would one want to start using a path p of some commodity j only after all other paths of commodity j get saturated. Furthermore, by the definition of IEWF, if a path p does not start with a split of 0, then its split can never become 0. Indeed, if p has a nonzero split at the beginning of an iteration, then it must carry some flow at the end of the iteration, and therefore the split of p in the following iteration cannot be 0. However, IEWF may converge to a UMMF flow with paths of 0 flow, and it is convenient for our proof of Step 3 that IEWF is defined also on splits

derived from flows with paths of 0 flow. (In such splits, a path with zero flow has a zero split.)

Our notion of EMMF multicommodity flow is define in terms of IEWF. Intuitively, a mulicommodity flow F is EMMF if, when we run IEWF with the splits derived from F , we get F . Since we want the family of EMMF multicommodity flows to be the same as the family of UMMF multicommodity flows, and UMMF multicommodity flows may have paths of 0 flow, then we have to define IEWF to work with zero splits that correspond to such paths.

4.2. Equilibrium Max-Min Fairness

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

Definition 4.1 (EMMF). A feasible multi-commodity flow is *EMMF* if, for every commodity j , the followings hold:

- (1) for every path $p \in P_j$, with $\lambda(p) > 0$ we get that $\tau(p) = v(f_j)$.
- (2) for every p with $\lambda(p) = 0$, $\tau(p) \leq v(f_j)$.

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 j all paths with nonzero flow are saturated exactly at $v(f_j)$. The paths of commodity j with 0 splits will be saturated by time $v(f_j)$. The following lemma follows from the definition of EMMF flow and from the definition of IEWF.

LEMMA 4.2. For each path $p \in P_j$ with $\lambda(p) > 0$, we have that $flow(p, v(f_j)) = \lambda(p)v(f_j)$. 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 4.3. 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 $e \in p$ 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 , then 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 $f_j \in L_i$ from the graph. It follows that we cannot increase the flow on any path of commodity i even if we remove all flows $f_j \in L_i$ from the graph, so F is upward max-min fair.

For the converse, assume that the flow F is upward max-min fair. Then by Claim 2.2, 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 follows that $\tau(p) \leq v(f_i)$ for any $p \in P_i$.

It is left to show that for every path $p \in P_i$, if $\lambda(p) > 0$, then $\tau(p) = \min_{e \in p} \tau(e) = v(f_i)$. We prove this by contradiction. Assume that there exists a path $p \in P_i$ with $\lambda_i(p) > 0$ and $\tau(p) < v(f_i)$. In particular, the set $K = \{p \mid \lambda(p) > 0, p \in P_j, \text{ and } \tau(p) < v(f_j)\}$ is not empty, since it contains p .

Let p' the path with minimum $\tau(p')$ among all the paths in K . Assume $p' \in P_\ell$. 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,

$$flow(e, \tau(p')) = \sum_{p|e \in p} \lambda(p)\tau(p') < \sum_j \sum_{p|e \in p, p \in P_j} \lambda(p)v(f_j) \leq c(e). \quad (1)$$

The strict inequality in Equation (1) holds, since (i) we have already proved that $\tau(e) = \tau(p') \leq v(f_j)$ for any j such that there exists a path p with $e \in p$ and $p \in P_j$ and (ii) we assume that $\tau(p') < v(f_\ell)$ and $e \in p'$. The last inequality in Equation (1) holds, since F is feasible. Equation (1) contradicts the assumption that e is saturated at time $\tau(p')$. $\square$

4.3. Convergence of IEWF

In this section, we show that the sequence of multicommodity flows computed by IEWF has 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 , and let $\lambda(p, t)$ be the split of path p at time t . Let g_j^0 be the flow of commodity j at the end of the previous iteration that defined the initial splits for the current iteration.

We define a flow function g_j^t by setting $g_j^t(p) = flow(p, t)$ if $p \in P_j$ is saturated at time t and $g_j^t(p) = g_j^0(p)$ if $p \in P_j$ is not saturated at time t . We define $v(g_j^t) = \sum_{p \in P_j} g_j^t(p)$. Thinking of $v(g_j^t)$ as a function of t , we note that it changes only when a path $p \in P_j$ is saturated, and therefore it is a step function. Its initial value, $v(g_j^0)$, is the total flow of commodity j in the previous iteration. Its final value is attained at time τ when the last path $p \in P_j$ is saturated and is equal to τ ($v(g_j^\tau) = \tau$), which is the total flow of commodity j in the current iteration.

Initially, at $t = 0$, for any path p we have $g_j^0(p) \geq flow(p, 0) = 0$. If $\lambda(p, t) > 0$, then the value $flow(p, t)$ is continuously increasing with t until the path p is saturated. At the time $\tau(p)$ when p is saturated, we set $g_j^{\tau(p)}(p) = flow(p, \tau(p))$. This decreases the value of p if $flow(p, \tau(p)) < g_j^0(p)$ and increases the value of p if $flow(p, \tau(p)) > g_j^0(p)$.

The following key lemma implies that for each commodity j there exists a time z_j , such that for all *unsaturated paths* $p \in P_j$ *simultaneously* we have that $flow(p, z_j) = g_j^0(p)$. It follows that a path $p \in P_j$ saturated before time z_j , satisfies that $flow(p, \tau(p)) < g_j^0(p)$. So when we set $g_j^{\tau(p)}(p) = flow(p, \tau(p))$, we decrease $v(g_j^t)$. A path $p \in P_j$ saturated after time z_j satisfies that $flow(p, \tau(p)) > g_j^0(p)$, so when we set $g_j^{\tau(p)}(p) = flow(p, \tau(p))$, we increase $v(g_j^t)$. Therefore, $v(g_j^t)$ as a function of t first decreases and then increases (see Figure 2).

LEMMA 4.4. *Consider some commodity j . For every unsaturated path $p \in P_j$, and any time t , we have that*

$$g_j^0(p) - flow(p, t) = \lambda(p, t)(v(g_j^t) - t). \quad (2)$$

In particular,

$$g_j^0(p) \geq flow(p, t) \iff v(g_j^t) \geq t.$$

Fig. 2. The value $v(g_j^t)$ as a function of time for some commodity j . It is a step function that first decreases and then increases. There are two times $\tau \leq \tau'$ where $v(g_j^t) = t$. At τ the flow along each unsaturated path p equals to $g_j^0(p)$ and at τ' the last unsaturated path of commodity j becomes saturated.

Lemma 4.4 also implies that time z_j is the first time when $v(g_j^{z_j}) = z_j$. Since $v(g_j^t)$ decreases up to time z_j , we have the following corollary.

COROLLARY 4.5. *If $v(g_j^t) \geq t$, then at any $\tau < t$ we have $v(g_j^\tau) \geq t$.*

PROOF OF LEMMA 4.4. To simplify the notation, we omit the subscript j from g throughout the proof. Clearly, the lemma holds at $t = 0$ by the definition of $\lambda(p, 0)$ and g^0 .

We first prove that if the lemma holds at a particular time t , then it continues to hold until the first time following t in which a path of commodity j saturates.

If no edge on an open path p of commodity j saturates in the next Δt time units following time t , then $v(g^{t+\Delta t}) = v(g^t)$ and $\text{flow}(p, t + \Delta t) = \text{flow}(p, t) + \lambda(p, t)\Delta t$ for any unsaturated path $p \in P_j$, so Equation (2) continues to hold at $t + \Delta t$ for all open paths of commodity j .

We now prove the lemma at the times in which some edge or path is saturated. Consider a set of edges E (on open paths of commodity j) that are saturated simultaneously at time t . To complete the proof we show that if the lemma holds right before the edges of E saturate then it continues to hold after the edges of E saturate.

Let t^- be the time right before the edges of E saturate and let t^+ be the time right after the edges of E saturate. We have to show that the lemma continues to hold at t^+ following the discrete changes in the splits and in g^t , that occur when E saturates.

Let p be a path which is still unsaturated at time t^+ . Notice that by the definition of g^t we have that $g^0(p) = g^{t^-}(p) = g^{t^+}(p)$.

Consider the following cases:

- (1) $\lambda(p, t^-) = \lambda(p, t^+) = 0$. Then we have that $g^0(p) = \text{flow}(p, t) = 0$, and Equation (2) holds for p at t^+ .
- (2) $\lambda(p, t^-) = 0$ but $\lambda(p, t^+) > 0$. Since $\lambda(p, t^-) = 0$ we have that $g^0(p) = 0$. By the definition of IEWF, since $\lambda(p, t^-) = 0$ and $\lambda(p, t^+) > 0$, it implies that all the paths with non-zero split are saturated by time t , and no path with zero initial split has any flow before time t . Hence $v(g^{t^+}) = t$ and $\text{flow}(p, t^+) = 0$. This implies that the identity holds at t^+ .

- (3) $\lambda(p, t^-) > 0$. If $g^{t^-}(p) = \text{flow}(p, t)$, then from the correctness of the lemma at t^- we have that $\lambda(p, t)(v(g^{t^-}) - t) = 0$. This implies, again from the correctness of the lemma at t^- , that for all open paths p at time t^- , $g^{t^-}(p) = \text{flow}(p, t)$, and, in particular, this holds for all paths that saturate at time t . Therefore, $g^{t^+}(p) = g^{t^-}(p)$ for all paths p that saturate at time t and $v(g^{t^-}) = v(g^{t^+})$. It follows that Equation (2) holds for p at t^+ .

Assume for the rest of the proof that $g^{t^-}(p) \neq \text{flow}(p, t)$ and $\lambda(p, t^-)0$.

Let Sat and UnSat be the set of saturated and unsaturated paths, respectively, of commodity j at time t^+ . By the definition of IEWF and, since we assume that $\lambda(p, t^-) > 0$, we have that

$$\lambda(p, t^+) = \frac{\lambda(p, t^-)}{\sum_{q \in P_j \cap \text{UnSat}} \lambda(q, t^-)}. \quad (3)$$

Since $g^{t^-}(p) \neq \text{flow}(p, t)$ from the correctness of the lemma at t^- , we have that $v(g^{t^-}) \neq t$, so $v(g^{t^-}) - t \neq 0$, and we can multiply the numerator and the denominator of Equation (3) by $(v(g^{t^-}) - t)$ and get that

$$\begin{aligned} \lambda(p, t^+) &= \frac{\lambda(p, t^-)(v(g^{t^-}) - t)}{\sum_{q \in P_j \cap \text{UnSat}} \lambda(q, t^-)(v(g^{t^-}) - t)} \\ &= \frac{g^{t^-}(p) - \text{flow}(p, t)}{\sum_{q \in P_j \cap \text{UnSat}} (g^{t^-}(q) - \text{flow}(q, t))}, \end{aligned} \quad (4)$$

where the last equality follows from the correctness of the lemma right before E is saturated. Consider the denominator of the last fraction in Equation (4). The following derivation shows that it is equal to $v(g^{t^+}) - t$.

Since the total flow of each commodity with unsaturated paths at time t is t , we get that:

$$\begin{aligned} v(g^{t^-}) - t &= \sum_{q \in P_j} (g^{t^-}(q) - \text{flow}(q, t)) \\ &= \sum_{q \in P_j \cap \text{Sat}} (g^{t^-}(q) - \text{flow}(q, t)) + \sum_{q \in P_j \cap \text{UnSat}} (g^{t^-}(q) - \text{flow}(q, t)). \end{aligned} \quad (5)$$

Let $\text{Sat}(t)$ be the set of paths that get saturated at time t . Since for every saturated path q that was saturated before time t , $g^{t^-}(q) = \text{flow}(q, t)$ we get that $\sum_{q \in P_j \cap \text{Sat}} (g^{t^-}(q) - \text{flow}(q, t)) = \sum_{q \in P_j \cap \text{Sat}(t)} (g^{t^-}(q) - \text{flow}(q, t))$. Furthermore, by the definitions of g^{t^-} and g^{t^+} , we have that

$$\begin{aligned} v(g^{t^+}) &= v(g^{t^-}) - \sum_{q \in P_j \cap \text{Sat}(t)} (g^{t^-}(q) - \text{flow}(q, t)) \\ &= v(g^{t^-}) - \sum_{q \in P_j \cap \text{Sat}} (g^{t^-}(q) - \text{flow}(q, t)). \end{aligned} \quad (6)$$

Substituting Equation (6) into Equation (5), we get that

$$v(g^{t^+}) - t = \sum_{q \in P_j \cap \text{UnSat}} (g^{t^-}(q) - \text{flow}(q, t)). \quad (7)$$

Now by substituting Equation (7) into Equation (4), and, since $g^0(p) = g^{t^-}(p) = g^{t^+}(p)$, we get that

$$g^{t^+}(p) - \text{flow}(p, t) = \lambda(p, t^+)(v(g^{t^+}) - t), \quad (8)$$

as required. $\square$

Consider an iteration of IEWF. Let $t_1, t_2, \dots, t_{m'}$ be the times in which edges get saturated during this iteration. Let $E(t_i)$ be the set of edges that are saturated at time t_i , and let $G^{t_i} = (g_1^{t_i}, g_2^{t_i}, \dots)$ be the multicommodity flow after the saturation of $E(t_i)$. Let G^{t_0} be the multicommodity flow at the beginning of the iteration.

We consider the sequence $G^{t_0}, G^{t_1}, \dots, G^{t_{m'}}$. The difference between G^{t_i} and $G^{t_{i+1}}$ is in the values of the flow along all paths that go through an edge of $E(t_{i+1})$ 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_0}).

Consider the sequence obtained from $G^{t_0}, G^{t_1}, \dots, G^{t_{m'}}$ 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 is saturated at t_{i+1} . We first 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 further break each increase (respectively, decrease) into multiple smaller increases (respectively, decreases) so the order of commodities does not change by each smaller increases (resp. decreases).

We show that this refined sequence of multicommodity flows is a *happy* sequence. To simplify the following presentation, we do not make the refinement of each transition from G^{t_i} to $G^{t_{i+1}}$ to many smaller single-path transitions explicit in our notation. We also abuse the notation slightly and refer to $G^{t_0}, G^{t_1}, \dots, G^{t_{m'}}$ as a *happy* sequence (meaning that its refinement is happy).

It then follows that the concatenation of the sequences $G^{t_0}, G^{t_1}, \dots, G^{t_{m'}}$ of all the iterations of IEWF is also a *happy* sequence, and therefore, by Corollary 3.2, the flow maintained by IEWF has a limit.

Furthermore, since the flow G^{t_0} at the beginning of each iteration of IEWF is feasible, it follows by Corollary 3.2 that the limit of IEWF is a feasible flow.

THEOREM 4.6. *For each iteration of IEWF the sequence $G^{t_0}, G^{t_1}, \dots, G^{t_{m'}}$ 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 appropriate single-path modifications, 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 $e_{i+1} \in E(t_{i+1})$ that saturates at time t_{i+1} . Let P^- be the set of paths through e_{i+1} that are saturated at t_{i+1} , and the flow through them in $G^{t_{i+1}}$ is smaller than the flow through them in G^{t_0} . Let P^+ be the set of paths through e_{i+1} that are 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} . (Note that we do not include in P^- paths through e_{i+1} that are saturated before t_{i+1} and the flow through them when they are saturated is smaller than the flow through them in G^{t_0} .)

Let $\Delta^- = \sum_{p \in P^-} \text{decrease}(p)$, where $\text{decrease}(p)$ is the amount by which the flow along p decreases. Let $\Delta^+ = \sum_{p \in P^+} \text{increase}(p)$, where $\text{increase}(p)$ is the amount by which the flow along p increases. We claim that $\Delta^+ \geq \Delta^-$.

The total flow along edge e_{i+1} at time t_{i+1} equals to the total flow through e_{i+1} at the end of the previous iteration, plus the sum of all increases minus the sum of all decreases in flow values of paths containing e_{i+1} , including decreases along paths saturated before t_{i+1} (not included in P^-). The sum of all the increases is simply Δ^+ , and the sum of all the decreases is at least Δ^- (since Δ^- includes only decreases in flow values of paths that saturate at time t_{i+1} and not before). The total flow on edge e_{i+1} at time t_{i+1} is its capacity, and hence it is at least as large as the total flow through e_{i+1} at the end of the previous iteration so we conclude that $\Delta^+ \geq \Delta^-$.

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^+ . The union of all these mappings for all edges $e_{i+1} \in E(t_{i+1})$ gives the function Π . Note that a path p can be counted as an increase in multiple edges $e_{i+1} \in E(t_{i+1})$ 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 units of decrease.

Assume that 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' is 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 4.4, since the flow on path p of commodity j decreases at t_{i+1} , we have that $v(g_j^{t_{i+1}}) \geq t_{i+1}$. It follow that also for every commodity β of rank larger than r at time t_{i+1} we have $v(g_\beta^{t_{i+1}}) \geq v(g_j^{t_{i+1}}) \geq t_{i+1}$. Since $t' < t_{i+1}$, we get from Corollary 4.5 that $v(g_\beta^{t'}) \geq t_{i+1} > t'$.

Since the flow on path p' of commodity j' increases at t' , Lemma 4.4 implies that $v(g_{j'}^{t'}) < t'$. Hence, all flows of rank at least r at time t_{i+1} satisfy $v(g_\beta^{t'}) \geq t_{i+1} \geq t' > v(g_{j'}^{t'})$ at time t' , and we get that $r' < r$. $\square$

4.4. The Limit of IEWF Is an 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 converges to a fixed point of f (EMMF) if f is continuous. Unfortunately, f is not continuous, and the following example shows that a tiny perturbation of initial conditions can cause IEWF to reach a very different outcome.

We have two commodities with the same source and sink. Commodity 1 has three paths: path A with capacity 100 and split 1, B with capacity 50 and split 0, and E with capacity 50 and split 0. Commodity 2 has a path C with split 1 and capacity 100, path D with capacity 50 and split 0, and shares E with Commodity 1, also with split 0. If we run IEWF on this instance, then we get that at time 100 both path A and C get saturated. Commodity 1 then continues with splits 1/2 and 1/2 on paths B and E , and Commodity 2 continues with split 1/2 and 1/2 on paths D and E . At time 150, path E would get saturated (25 units from Commodity 2 and 25 units from Commodity 1). Commodity 1 would then continue to fill path B , and Commodity 2 would fill path D . Eventually, both commodities would route 175 units of flow.

Now perturb the split of Commodity 1 on path A to $1 - \epsilon$ and on path B to ϵ . Now, at time 100, Commodity 2 would change the splits on D and E to be 1/2 and 1/2. At time $\frac{100}{1-\epsilon}$, Commodity 1 saturates path A and routes all of its flow on B (with split 1). At time 150, Commodity 1 saturates path B ; at the same time (150), Commodity 2 routes 100 on C , 25 on D , and 25 on E . At time 150, Commodity 1 changes the split on path E from 0 to 1. At time 166.66, E will get saturated. Commodity 2 routes 33.66 units of flow on E , and Commodity 1 routes 16.33 units of flow on E . Now all paths of commodity 1 are

saturated, and its value is 166.66. Commodity 2 will continue to fill up path D , and its value will be 183.33. That is, we perturbed the splits by ϵ , but the resulting flow changed by a large quantity.

Despite this discontinuity, Theorem 4.7 shows that IEWF converges to an EMMF flow. The proof of this theorem is by contradiction. We show that if the limit (which exists as we proved in the previous section) is not EMMF, then when the flow of IEWF is close to it, it must incur a large change.

THEOREM 4.7. *The limit F^* of the sequence of flows $F^1, F^2, \dots$, produced by the iterations of IEWF is an EMMF multicommodity flow.*

PROOF. Let $F^* = (f_1^*, \dots, f_k^*)$ and $F^i = (f_1^i, \dots, f_k^i)$. From the definition of the limit, we have that for every ϵ , there is an i_ϵ such that for any $i \geq i_\epsilon$, any commodity j and any path $p \in P_j$ (recall that P_j is the set of all paths of commodity j) we have $|f_j^i(p) - f_j^*(p)| \leq \epsilon$.

We would like to show that if F^* is not EMMF, then for some sufficiently small ϵ , for any i_ϵ , there exists $i \geq i_\epsilon$, a commodity j , and a path $p \in P_j$ such that $|f_j^i(p) - f_j^*(p)| > \epsilon$. This will be a contradiction to the assumption that F^* is the limit.

Let $p \in P_j$ be a path of commodity j , and let $\lambda^*(p) = f_j^*(p)/v(f_j^*)$. Let R^* be the run of IEWF with the initial splits λ^* . Let $t_1^* < t_2^* < \dots$ be the saturation times of the edges in R^* .

Since F^* is not EMMF, we know that there exists a commodity C such that in R^* the paths of C are not saturated at the same time. Let t_c^* be the *first* time in which we have a commodity C such that either of the following occurs:

- (1) Some paths p of C with $\lambda(p) > 0$ get saturated at t_c^* and some other paths remain open.
- (2) All paths p of C with $\lambda(p) > 0$ get saturated at t_c^* , but C still has open paths with $\lambda(p) = 0$ that change their split to a positive value at t_c^* .

In other words, R^* behaves as an EMMF up to time t_c^* : At each time t_s^* , $s < c$, if a commodity C' has a path p with $\lambda(p) > 0$ that is saturated then all paths p of C' with $\lambda(p) > 0$ are saturated at t_s^* . Furthermore, all paths p of C' with $\lambda(p) = 0$ are saturated at time t_ℓ^* for some $\ell < s$.

Let τ^* be the time when the last path p of commodity C with $\lambda^*(p) > 0$ saturates in R^* , and recall that by the definition of IEWF, $v(f_C^*)$ is the time when the last path p of commodity C saturates in R^* . With this notation when Case 2 occurs, we have that $t_c^* = \tau^* < v(f_C^*)$ and when Case 1 occurs we know that $t_c^* < \tau^*$.

Fix some very small $\epsilon > 0$, much smaller than $v(f_C^*) - t_c^*$ (which is strictly positive by our choice of C and t_c^*) and any other parameter of R^* . Fix also some tentative i_ϵ . Recall that we want to show that there exists $i \geq i_\epsilon$ and a path $p \in P_j$ such that $|f_j^i(p) - f_j^*(p)| > \epsilon$.

Let R^i be the run of IEWF that produces F^i , starting with splits $\lambda^i(p) = f_j^{i-1}(p)/v(f_j^{i-1})$ for every $p \in P_j$. Analogously to τ^* , let τ^i be the time when the last path p of commodity C for which $\lambda^*(p) > 0$ saturates in R^i , and we recall that, by the definition of IEWF, $v(f_C^i)$ is the time when the last path p of commodity C saturates in R^i .

Let $i > i_\epsilon$ large enough. If for some commodity j , $|v(f_j^i) - v(f_j^*)| > \epsilon|P_j|$, then for some $p \in P_j$ we have that $|f_j^i(p) - f_j^*(p)| > \epsilon$, and we get a contradiction. Therefore, we can assume that

$$|v(f_j^i) - v(f_j^*)| < \epsilon|P_j| = O(\epsilon). \quad (9)$$

Furthermore, since $i > i_\epsilon$, we also get a contradiction if $|v(f_j^{i-1}) - v(f_j^*)| > \epsilon|P_j|$, so we can also assume that

$$|v(f_j^{i-1}) - v(f_j^*)| < \epsilon|P_j| = O(\epsilon). \quad (10)$$

From Equations (9) and (10), we get that

$$|v(f_j^i) - v(f_j^{i-1})| = O(\epsilon). \quad (11)$$

From Equation (10), we must have that

$$|\lambda^i(p) - \lambda^*(p)| = O(\epsilon), \quad (12)$$

for every commodity j and path $p \in P_j$, as otherwise we immediately get that $|f_j^{i-1}(p) - f_j^*(p)| = |\lambda^i(p)v(f_j^{i-1}) - \lambda^*(p)v(f_j^*)| > \epsilon$, which is a contradiction.

For each path p , let $\lambda^*(p, t)$ be the split of p right after time t in R^* , and let $\lambda^i(p, t)$ be the split of p right after time t in R^i (i.e., in case R^* changes the split of p at t , then $\lambda^*(p, t)$ is the split after the change, and the same holds for $\lambda^i(p, t)$ and R^i .)

For each edge e , let $r^*(e, t)$ be the residual capacity of e at time t in R^* , and let $r^i(e, t)$ be the residual capacity of e at time t in R^i . For each path p , let $f^*(p, t)$ be flow along p at time t in R^* , and let $f^i(p, t)$ be the flow along p at time t in R^i .

Let $E(t_s^*)$ be the set of edges that get saturated at time t_s^* in R^* , and let $P(t_s^*)$ be the set of paths of all commodities (of zero and non-zero splits) that go through $E(t_s^*)$ and get saturated at t_s^* in R^* . Let $\hat{t}_s^i$ be the time in which the first path in $P(t_s^*)$ is saturated in R^i , and let t_s^i be the time in which the last path in $P(t_s^*)$ is saturated in R^i .

The following claims follow for $s < c$ by induction on s using Equation (12) for the base case. The constant hidden in the $O(\epsilon)$ terms depends on s . The first two claims also hold for $s = c$. Intuitively, these claims show that, since R^i and R^* start with approximately the same splits by Equation (12), then their sets of unsaturated paths are the same up to time t_s^* , and the splits of these paths are similar.

- (1) $|\hat{t}_s^i - t_s^*| = O(\epsilon)$ and $|t_s^i - t_s^*| = O(\epsilon)$.
- (2) For any path p , $|f^i(p, t_s^i) - f^*(p, t_s^*)| = O(\epsilon)$ and thereby for any edge e , $|r^i(e, t_s^i) - r^*(e, t_s^*)| = O(\epsilon)$.
- (3) For any path p that is still open right after time t_s^i in R^i , we have that $|\lambda^i(p, t_s^i) - \lambda^*(p, t_s^*)| = O(\epsilon)$.

Claim 2 implies that a path that is open in t_s^* in R^* is open in t_s^i in R^i and has about the same residual capacity. We do not use Claim 3 in the sequel, but it is essential for the induction that proves the claims. We give this induction at the end of the proof.

During the time interval $[\tau^i, v(f_C^i)]$, only paths $p \in P_C$ with $\lambda^*(p) = 0$ are open. If $v(f_C^i) - \tau^i > \epsilon|P_C|$, then we have that some path $p \in P_C$ with $\lambda^*(p) = 0$ has flow larger than ϵ in F^i , that is, there exists a path $p \in P_C$ such that $f_C^i(p) - f_C^*(p) = f_C^i(p) > \epsilon$, which gives a contradiction. So from now on we assume that

$$v(f_C^i) - \tau^i < \epsilon|P_C| = O(\epsilon). \quad (13)$$

Equipped with these assumptions, we continue according to the cases defined above.

First, consider Case 2. That is, $t_c^* = \tau^*$. We already assumed that $|v(f_C^i) - v(f_C^*)| = O(\epsilon)$ (Equation (9)) and that $|v(f_C^i) - \tau^i| = O(\epsilon)$ (Equation (13)). Claim 1 guarantees that $|t_c^i - t_c^*| = O(\epsilon)$ and that $|t_c^i - \tau^i| = O(\epsilon)$. So it follows that

$$\begin{aligned} v(f_C^*) - t_c^* &\leq |v(f_C^*) - v(f_C^i)| + |v(f_C^i) - \tau^i| + |\tau^i - t_c^i| + |t_c^i - t_c^*| \\ &= O(\epsilon). \end{aligned}$$

This contradicts our choice of ϵ .

Now consider Case 1. That is, $t_c^* < \tau^*$. Let p be a last path with $\lambda^*(p) > 0$ that saturates in R^i at time τ^i . By the definition of IEWF,

$$\begin{aligned} f_C^i(p) &\geq \lambda^i(p)t_c^i + \lambda^i(p, t_c^i)(\tau^i - t_c^i) \\ &= \lambda^i(p)v(f_C^i) + (\lambda^i(p, t_c^i) - \lambda^i(p))(v(f_C^i) - t_c^i) - \lambda^i(p, t_c^i)(v(f_C^i) - \tau^i). \end{aligned} \quad (14)$$

We now analyze the three terms in the right-hand side of Equation (14). For the first term, we have by the definition of $\lambda^i(p)$ and by Equation (11) that

$$\lambda^i(p)v(f_C^i) = \lambda^i(p)v(f_C^{i-1}) + \lambda^i(p)(v(f_C^i) - v(f_C^{i-1})) = f_C^{i-1}(p) \pm O(\epsilon).$$

Let $p' \in P_C$ be a path that saturates at t_c^* in R^* . By Claim 1 (and the choice of i), p' saturates by time t_c^i in R^i . Furthermore, by Claim 2 and, since $t_c^* < \tau^*$, it follows that $t_c^i < \tau^i$. So p stays open in R^i following time t_c^i . Therefore, the split of p in R^i increases by a factor of at least $1/(1 - \lambda^i(p'))$ following t_c^i . So for the second term we have

$$(\lambda^i(p, t_c^i) - \lambda^i(p))(v(f_C^i) - t_c^i) \geq \lambda^i(p) \frac{\lambda^i(p')}{1 - \lambda^i(p')} (v(f_C^i) - t_c^i). \quad (15)$$

Since $|v(f_C^i) - v(f_C^*)| = O(\epsilon)$ by Equation (10), $|\lambda^i(p) - \lambda^*(p)| = O(\epsilon)$ and $|\lambda^i(p') - \lambda^*(p')| = O(\epsilon)$ by Equation (12), and $|t_c^i - t_c^*| = O(\epsilon)$ by Claim 1, we get that the right-hand side of Equation (15) is at least as large as

$$(\lambda^*(p) - O(\epsilon)) \frac{\lambda^*(p') - O(\epsilon)}{1 - \lambda^*(p') + O(\epsilon)} (v(f_C^*) - t_c^* - O(\epsilon)) > \eta,$$

where $\eta > 0$ is some constant. For the third term, by Equation (13), we have

$$\lambda^i(p, t_c^i)(v(f_C^i) - \tau^i) = O(\epsilon).$$

Summing up the bounds on the three terms of the right-hand side of Equation (14), we get that for a sufficiently small ϵ ,

$$f_C^i(p) - f_C^{i-1}(p) \geq \eta - O(\epsilon) > 0,$$

which is a contradiction.

It remains to establish Claims 1–3 stated above. We do that by induction on s . We assume that the statement is true for some $s < c$ and prove that it holds also for $s + 1$.

By Claims 1 and 2 of the induction assumption, the same set of paths are saturated following time t_s^* in R^* and following time t_s^i in R^i . Furthermore, by the induction assumption, we have that for any edge e

$$\left| \frac{r^i(e, t_s^i)}{\Lambda^i(e, t_s^i)} - \frac{r^*(e, t_s^*)}{\Lambda^*(e, t_s^*)} \right| = O(\epsilon), \quad (16)$$

where $\Lambda^*(e, t)$ is the sum of $\lambda^*(p, t)$ of all unsaturated paths p that go through e right after time t in R^* and $\Lambda^i(e, t)$ is the sum of $\lambda^i(p, t)$ of all unsaturated paths p that go through e right after time t in R^i .

This implies the following (in R^i):

- (1) The next edge e' that is saturated following time t_s^i is an edge of $E(t_{s+1}^*)$.
- (2) The time, $\hat{t}_{s+1}^i$, in which e' saturates, is within an additive factor of $O(\epsilon)$ from t_{s+1}^* .
- (3) We have that for every path p , $|f^i(p, \hat{t}_{s+1}^i) - f^*(p, t_{s+1}^*)| = O(\epsilon)$ and for any edge e , $|r^i(e, \hat{t}_{s+1}^i) - r^*(e, t_{s+1}^*)| = O(\epsilon)$.

It follows that all the paths of $P(t_{s+1}^*)$ are saturated within the next $O(\epsilon)$ time units following $\hat{t}_{s+1}^i$. So Claim 1 holds for $s+1$. Claim 2 also holds for $s+1$, since the flow on any path cannot change by more than α in α time units. In our case, $t_{s+1}^i - \hat{t}_{s+1}^i = \alpha = O(\epsilon)$.

Assume that $s+1 < c$ and consider the time, t_{s+1}^i , in which all paths of $P(t_{s+1}^*)$ are saturated. Let j be a commodity that still has open paths at t_{s+1}^i in R^i . Then, by the definition of c , the only paths of commodity j that are saturated at time t_{s+1}^* in R^* are paths $p \in P_j$ with $\lambda^*(p) = 0$. Let Z be the set of these paths. It follows from Claims 1 and 2, which we already established for $s+1$, that the paths of commodity j that are saturated in the time interval $[\hat{t}_{s+1}^i, t_{s+1}^i]$ in R^i are the paths in Z . By the induction assumption (Claim 3 for s), the splits $\lambda^i(p, t_s^i)$, $p \in Z$, are $O(\epsilon)$. So when paths of Z saturate, splits of paths in $P_j \setminus Z$ can change only by $O(\epsilon)$ and thereby Claim 3 follows for $s+1$. $\square$

5. EXPERIMENTAL STUDY

We analyzed the behavior of IEWF on two types of networks: a metro level abstraction of the Google backbone network [Jain et al. 2013] and a set of synthetic networks generated using the Waxman model [Waxman 1988] in varying sizes. 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 Zegura et al. [1996]. These parameters give 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 that 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 of n nodes.

As explained in the Introduction, the GMMF multicommodity flow 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 performance of IEWF.

5.1. IEWF Split Selection and Convergence

In our first experiment, we studied the convergence rate of IEWF and the affect of the initial split selection on the performance of IEWF and its convergence rate.

We compared 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 *length exponential decay*, uses splits proportional to $1/10^\ell$, where ℓ is the number of hops in the path. This method prefer paths with fewer hops by giving them a larger 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, then order them arbitrarily. 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*.

To estimate the number of iterations required for convergence, we tracked the vector of differences between the flow values of the different commodities at consecutive

(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. 3. Experimental results.

iterations. When the norm of this vector becomes negligible, then we conclude that the algorithm converged.

The relative fraction of each commodity from the total throughput compared to its fraction in the GMMF solution was only slightly affected by the initial splits (see Section 5.2). This justifies considering the global throughput as a measure by which we compared the affect of different initial splits. Although IEWF changes splits in every iteration, the initial splits may still have a significant affect on the results for two main reasons: (1) When the splits change, they change proportionally to their previous values, and (2) as we see below, IEWF converges within a very small number of iterations.

Figure 3(a) depicts the ratio between the overall throughput obtained by the IEWF algorithm after 2 and 10 iterations, and the total throughput achieved by the GMMF multicommodity flow for the same input. Each point in the graph is the average of 50 runs, each having a random subset of a set 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.

The experiments show that after fewer than 10 iterations IEWF converged. Subsequent iterations change the vector of flow values by a negligible amount. In fact, Figure 3(a) shows that IEWF already collected most of its final throughput after 2 iterations. This is mainly due to few commodities of high flow values. But by inspecting the changes in the vector of flow values more carefully, we see that there is still a difference

between runs of 2 and 10 iterations (which is not reflected by the total throughput) in the fairness between commodities. A run of 10 iterations balances better the commodities in the sense that commodities of similar flow value in the GMMF solution have closer values after 10 iterations than after 2 iterations.

It is clear from the graph that the split selection rules that take into account the length of the paths have higher overall throughput. The reason is that fewer links are used when shorter paths are preferred, and the network, in general, is less congested, and 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 a path only when all the shorter paths of its 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.”

We also conducted this experiment on the family of Waxman graphs. The results are shown in Figure 3(b). The x -axis represents the number of nodes in the graph, and the y -axis is the average of the throughput of the UMMF flow divided by the throughput of the GMMF flow on all the graphs that we tested. We used exponential decay as the split selection and compared the results after 2 and 10 iterations. On these graphs, the UMMF solution reached at least 96% of the throughput of the GMMF flow.

Similarly 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 two-connected. For small graphs, this means that we had to reject many of the graphs that we sampled, and this may have generated a skew in the distribution.

5.2. Allocations of Individual Commodities

Figure 3(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 from 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, the 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 3(c) in the practical scenarios we considered, the results of the GMMF multicommodity flow and the UMMF multicommodity flow 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 make much difference, so we do not show the results for 50 iterations.

5.3. IEWF Stability

Finally, we tested the stability of the two approaches on the Google backbone network to see how sensitive the resulting multicommodity flow is to a small change in the demands. In real life, demands vary over time (usually in a rather smooth manner), which may lead to changes in the Traffic Engineered (TE) routing (paths + splits) deployed in the network. An important property of any TE algorithm is stability when reacting to these changes, as deploying new splits or paths in 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 the variance of these splits. The results are presented in Figure 3(d): The top line in the graph represents a histogram of the variance of the splits in the GMMF solution, and the other two lines represent a histogram of the variance of the splits in the IEWF runs. (More specifically, we sort the paths by increasing variance and plot the variance for each path in this order.) The variance in the splits generated by IEWF was significantly lower than the variance of the splits in the GMMF solution. One can conclude that a small change in the set of demands led to a rather big change in the GMMF solution, while the IEWF solution was more stable.

REFERENCES

- Y. Afek, Y. Mansour, and Z. Ostfeld. 1996. On the convergence of rate based flow control. In *STOC*. 106–143.
- M. Allalouf and Y. Shavitt. 2008. Centralized and distributed algorithms for routing and weighted max-min fair bandwidth allocation. *IEEE/ACM Trans. Netw.* 16, 5 (Oct. 2008), 1015–1024. DOI: <http://dx.doi.org/10.1109/TNET.2007.905605>
- D. Bertsekas and R. Gallager. 2001. *Data Networks*. Prentice-Hall, Englewood Cliffs, NJ.
- Y. Dinitz. 2006. Dinitz' algorithm: The original version and Even's version. In *Essays in Memory of Shimon Even*. 218–240.
- L. K. Fleischer. 2000. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discr. Math.* 13 (2000), 505–520.
- N. Garg and J. Könemann. 2007. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.* 37, 2 (2007), 630–652.
- R. Jain, D. Chiu, and W. Hawe. 1998. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *Computing Research Repository* cs.NI/9809 (1998).
- Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2013. B4: Experience with a globally-deployed software defined wan. *SIGCOMM Comput. Commun. Rev.* 43, 4 (2013), 3–14.
- D. Nace and M. Pioro. 2008. Max-min fairness and its applications to routing and load-balancing in communication networks: A tutorial. *IEEE Commun. Surv. Tutor.* 10, 4 (2008), 5–17.
- B. Radunovic and J. Le Boudec. 2002. A unified framework for max-min and min-max fairness with applications. In *40th Annual Allerton Conference on Communication, Control, and Computing*.
- N. Wang, K. Ho, G. Pavlou, and M. Howarth. 2008. An overview of routing optimization for internet traffic engineering. *IEEE Commun. Surv. Tutor.* 10, 1 (2008), 36–56. DOI: <http://dx.doi.org/10.1109/COMST.2008.4483669>
- B. M. Waxman. 1988. Routing of multipoint connections. *IEEE J. Select. Areas Commun.* 6, 9 (1988), 1617–1622.
- E. Zegura, K. Calvert, and S. Bhattacharjee. 1996. How to model an internetwork. In *INFOCOM*. 594–602.
- Y. Zhou. 2003. *Resource Allocation in Computer Networks: Fundamental Principles and Practical Strategies*. Ph.D. Dissertation. Drexel University.

Received January 2015; revised February 2016; accepted October 2016