Centralized and Distributed Algorithms for Routing and Weighted Max-Min Fair Bandwidth Allocation

Miriam Allalouf*, Yuval Shavitt*

Abstract

Given a set of demands between pairs of nodes, we examine the traffic engineering problem of flow routing and fair bandwidth allocation where flows can be split to multiple paths (e.g., MPLS tunnels). This paper presents an algorithm for finding an optimal and global per-commodity max-min fair rate vector in a polynomial number of steps. In addition, we present a fast and novel distributed algorithm where each source router can find the routing and the fair rate allocation for its commodities while keeping the locally optimal max-min fair allocation criteria. The distributed algorithm is a fully polynomial epsilon-approximation (FPTAS) algorithm and is based on a primal-dual alternation technique. We implemented these algorithms to demonstrate its correctness, efficiency, and accuracy.

I. INTRODUCTION

Traffic engineering is a paradigm where network operators control the traffic and allocate resources in order to achieve goals, such as, maximum flow or minimum delay. One challenge is to allow different flows to share the network, so that the total flow will be maximized while preserving fairness.

We consider as input a network topology and directional link capacities, a list of ingress-egress pairs, and per-pair traffic demand. This list of demands may represent aggregates of (e.g., TCP) connections, such as client traffic (university campus, business client) and will typically be expressed by average or maximum required rates. Thus, traffic between an ingress-egress pair may be split arbitrarily among different paths without causing packet reorder in the connections comprising each demand. Our goal is to fulfill clients' demands while sharing the network bandwidth fairly. This is facilitated by laying the set of paths to be used between each pair in the network, and by allocating them bandwidth using the weighted max-min fairness criterion.

* The authors are with the School of Electrical Engineering, Department of Electrical Engineering-Systems, Tel-Aviv University. e-mail: miriama@eng.tau.ac.il, shavitt@eng.tau.ac.il. Corresponding author: Miriam Allalouf.

One way to maximize the network flow is to formulate the problem as a Maximum Multi-Commodity Flow (*MMCF*) problem which can be solved using linear programming (LP). While the solution will maximize the flow, it will not always do it in a fair manner. Flows that traverse several congested links will be allocated very little bandwidth or none at all, while flows that traverse short hop distances will receive a large allocation of bandwidth. In an attempt to introduce fairness into the maximum flow problem, the Maximum Concurrent Multi-Commodity Flow (*MC_MCF*) problem was suggested [1]. In this *MC_MCF* LP formulation, we are given a set of K demands dem_i, one per each commodity pair (s_i, t_i) and are required to satisfy the maximum equal fraction λ of all demands and to seek a routing that maximizes network flow. However, the achieved solution under-utilizes the network, sometimes saturating only a small fraction of its links.

The max-min fair allocation strikes a balance between fairness and the need to fully utilize the network. An allocation of bandwidth or rates to a set of connections is said to be max-min fair if it is not possible to increase the allotted rate of any connection while decreasing only the rates of those connections which have larger rates. An extended version of the max-min fair allocation problem is the variable-routing scenario, where the flow between two terminals (a commodity) may be split among several paths and the set of the paths that achieve such maximum fair allocation is not part of the input. The variable-routing version, studied in this paper, is more difficult than the classical max-min fair allocation problem since it couples the bandwidth allocation and the flow routing problem. There can be more than one weighted max-min fair rate vector in a variable-routing scenario, but only one is the global maximum max-min rate vector. The globally weighted max-min fair rate vector is the lexicographically largest feasible vector. This is different from the case when the input paths are fixed and only a single path exists between the source and the destination. In this case there is a unique max-min fair rate vector and this vector is lexicographically the largest rate vector.

The Max-min fairness bandwidth allocation criterion was mostly defined in the context of one fixed path per

session, where a session is defined by a pair of terminals. A simple algorithm that finds the max-min fair allocation where the routes are given appears in Bertsekas [2].

Many distributed algorithms were suggested for the dynamic adjustments of flow rates to maintain max-min fairness when single routes are given [3], [4], [5], [6]. The above algorithms differ by the assumptions on the allowed signaling, and available data. Bartal *et al.* [6] found the total maximum flow allocation in a network for given routes using distributed computations as the input to the global *MMCF* LP problem. Kelly *et al.* [7] proposed the proportional fairness concepts and a convergence algorithm. Mo and Warland [8] generalized proportional fairness and suggested end-to-end flow control for TCP streams by changing the transmission window size, but again they deal with flow allocation without routing.

The max-min variable-routing scenario was rarely discussed. Chen and Nahrstedt [9] provided max-min fair allocation routing. They present an un-weighted heuristic algorithm that selects the best single path so the fairness-throughput is maximized upon an addition of a new flow. Their algorithm searched this route out of the possible paths for each new flow. Kleinberg *et al.* [10] provided an interesting introduction regarding the relationship between the way in which one selects paths for routing and the amount of throughput one obtains from the resulting max-min fair allocation on these paths. Megido [11] addressed this problem for a single commodity maximum flow. He showed that the fairest flow is the maximum flow, namely throughput is not sacrificed by imposing fairness. Kleinberg *et al.* [10] found approximated routing to provide a max-min bandwidth allocation for single source unsplitable flow routing. Goel *et al.* [12] developed an on-line algorithm that finds the max-min fair vector in an $O(\log^2 n \log^{1+\varepsilon} U/\varepsilon)$-competitive ratio, where routes are given; Buchbinder and Naor [13] improved the bounds in [12]. In the wireless context, Hou et al. [14] present an algorithm that finds the optimal weighted max-min fair rate vector in a variable routing formulation using an LP solver like we do, though with a different saturation test.

Our work focuses on the variable-routing weighted max-min fair allocation problem and finds an off-line algorithm for calculating the global weighted max-min fair rate vector in a polynomial number of steps. In addition, we present a fast distributed algorithm for the locally optimal weighted max-min fair rate vector that extend and embed the variable-size increments techniques. These techniques were never used before for routing and bandwidth allocation using the weighted max-min criteria, nor they were used in a distributed

manner. In extensive simulations, our algorithm, which avoids using an LP solver, is shown to reach a solutions up to an ε from the optimal.

Our off-line algorithm, called *OPT_WMMF* (Optimal Weighted Max-Min Fair multi-Commodity)[1], finds the optimal per-commodity max-min fair rate vector. It solves iteratively a re-formulated *MC_MCF* LP until network saturation is achieved. Each iteration may change the routing selected in the previous iteration but may not decrease the already allocated per-commodity bandwidth. As a result, it is hard to distribute this algorithm. In addition, LP solvers have relatively long (though still polynomial) running time, which may make them impractical for large networks.

Thus, we developed a centralized algorithm, ϵ_*WMMF*, that converges to a maximal (but not necessarily maximum) weighted max-min fair solution and can be distributed (ϵ_*WMMFdist*). This family of algorithms relies on the idea of embedding the *MC_MCF* solution into the process of finding the rate vector of the max-min fair flows and uses its plain formulation, and provide a per-commodity weighted max-min fair rate vector, though we can not guarantee it will be the optimal vector. ϵ_*WMMF* and ϵ_*WMMFdist* are centralized and distributed versions of an FPTAS approximation to the *LOC_WMMF* algorithm. They run on the dual problem to the *MC_MCF* and enable a more efficient centralized algorithm and consequently, the distributed algorithm[2].

Maximum Concurrent Multi-Commodity Flow Problem Related Works. Most of the studies that combined the LP formulation for the traffic engineering design chose a multi-commodity flow formulation that considers the demands but they do not discuss the max-min fairness in conjunction with maximum throughput as our *LOC_WMMF* algorithm does[3]. A few directions for building approximation algorithms for the *MMCF* problem were suggested in the past. Young [17] described a random algorithm that computes the flow by solving a shortest path problem (on the dual LP) and pushing one unit of flow over it, at each step. Garg and Könemann [18] using detailed analysis extended Young's algorithm and improved its time complexity by pushing enough flow so as to saturate the bottleneck link of the path.

---

[1]For abbreviations see Table I.

[2]An approximation scheme is an algorithm that computes a solution within a factor of $1 - \epsilon$ of the optimal for any constant $\epsilon > 0$. The approximation scheme is a fully polynomial time approximation (FPTAS) if its running time is polynomial in both $1/\epsilon$ and the problem input size.

[3]Relevant mathematical and algorithmic background on *MC_MCF* problem and its complexity can be found in [1], [15], [16]

MMCF	Maximum Multi Commodity Flow problem
MC_MCF	Maximum Concurrent Multi Commodity Flow problem
OPT_WMMF	Optimal Weighted Max-Min Fair multi-Comm. Alg.
LOC_WMMF	LOCally-optimal Weighted Max-Min Fair multi-Commodity Alg.
ϵ_*WMMF*	Approx. Weighted Max-Min Fair Alg.
ϵ_*WMMFdist*	Aapprox. Optimal Weighted Max-Min Fair Dist. Alg.

TABLE I

PROBLEM AND ALGORITHM NAME ABBREVIATION LIST

Fleischer [19] and Karakostas [20] improved the *MMCF* approximation algorithm by partitioning their technique into phases and by re-calculating a set of shortest paths for all the commodities with the same source node, instead of per commodity as done before, and reduced the dependence of the running time on the number of the commodities, K, to a logarithmic factor.

Our ϵ_*WMMF* algorithm uses and extends the variable-size increments techniques (presented by Garg and Könemann [18] and Fleischer [19]) to achieve a new solution to the max-min fair. These algorithms do not deal with explicit net flows per path, thus, to achieve network saturation using the dual problem, we extend their technique using deeper understanding of the trade-off between the network saturation and link length assignments. The distributed algorithm, ϵ_*WMMFdist*, provides a mechanism where each source node can maximally and efficiently allocate bandwidth to its own clients, supply them a routing and still guarantee global fairness.

The following section explains the max-min fairness criteria in our context; states definitions; and explains the theoretical tools we use in the paper, namely, the primal *MC_MCF* problem and its dual problem. Section III presents the iterative algorithm that finds the globally optimal max-min fair rate-vector. Section IV describes our *LOC_WMMF* family of algorithms, including the distributed algorithm. Finally we conclude the paper.

II. DEFINITIONS, MODEL, AND TOOLS

A. Max-Min fairness

To clarify the differences between fairness criteria and algorithms, consider the example in Fig. 1, which depicts a line network with four nodes which are connected by one unit capacity links. Four flows demands are depicted in the figure each with a unit demand. Note that in this example, there is only a single path between each pair of nodes, thus only single bandwidth allocation is considered. The *MMCF* problem results in an allocation vector (0,1,1/2,1/2) starving flow 1 since it passes through two congested links. The total flow this allocation achieves is the maximum possible, 2 units. The max-min fair [2]

vector in this case is (1/3,2/3,1/3,1/3) which achieves a flow of 5/3. The weighted max-min fair vector, treating each flow in the example as a commodity (since the source-destination pairs are different), when all the weights are equal, is the same rate vector. In case pair 1 is given a double weight than the rest of the nodes, it will be allocated double the bandwidth in its bottleneck link (link(2,3)) and the weighted max-min fair vector is (1/2,1/2,1/4,1/4).
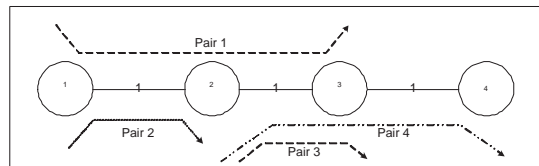

Fig. 1. Example of flows assignment

The max-min fair definition is traditionally stated for the case where each flow takes a pre-assigned single path [2]. A variable-routing scenario is a more complicated problem that requires to find both the routing and the bandwidth allocation such that fairness will be achieved, given only the network topology and link capacities. The optimal solution in the variable routing scenario is the set of paths that achieves the fairest bandwidth allocation, and the per commodity allocation. Our framework considers a flow that may also be split among several paths. More than that, this scheme is extended to consider weights for the flows. We state it in the following definitions.

Definition 1 *The Commodity Rate Vector, ρ, is a vector whose elements are the rates which were assigned to the commodities (source-destination pairs).*

From this definition we can write $\sum_{P_{ij} \in P_i} f(P_{ij}) = \rho_i$ where P_i is the set of all the paths that are assigned to commodity i, and $f(P_{ij})$ is the flow of commodity i along its j's path. We denote by ρ^* the optimal weighted max-min fair commodity rate vector, by ρ_i^* its ith element, by f_i the flow rate vector, and by dem_i the demand or the weight of commodity i.

Definition 2 *The vector ρ is said to be (**weighted**) max-min fair if it is feasible and if each of its elements ρ_i cannot be increased without decreasing any other element ρ_k for which $(\rho_i/\mathbf{dem_i} \geq \rho_k/\mathbf{dem_k}) \rho_i \geq \rho_k$*

The two definitions above also hold when traffic may be split to several paths.

Definition 3 *Given network topology G and its link capacities, a variable-routing max-min fair bandwidth allocation problem achieves a max-min fair solution by*

choosing the set of paths between source-destination pairs (i.e., the paths are not part of the input).

A variable-routing scenario can find multiple weighted max-min fair rate vector, since it can consider various paths in each selection such that the obtained rate allocation is different. Only one rate vector is the optimal or the global maximum max-min rate vector.
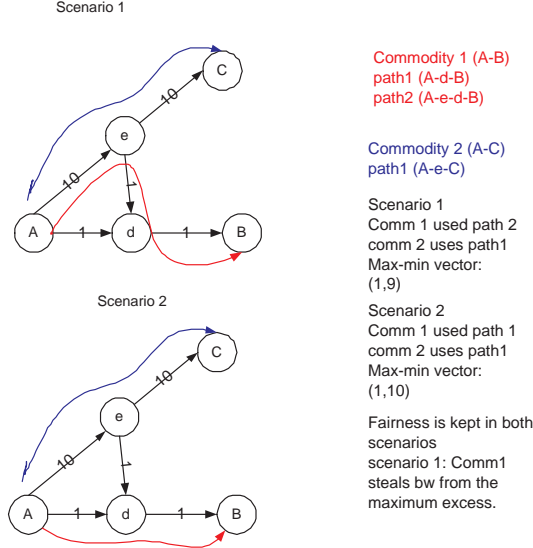

Fig. 2. An example of two routing layouts, each assigns weighted max-min fair rate vector $(1, 9)$ and $(1, 10)$ respectively. The globally weighted max-min fair rate vector is provided by scenario 2 since $(1, 10) >_{lexi} (1, 9)$

Definition 4 *Given network topology G, its link capacities, and, CR, the set of the rate vectors that are weighted max-min fair for the list of commodities; The globally weighted max-min fair rate vector, ρ^* is the lexicographically largest feasible vector among CR's vectors*[4].

Fig. 2 presents an example for definition 4.

Definition 5 *The Commodity Rate Vector, ρ, is αC coordinate-wise competitive if $\forall i$ the $i'th$, coordinate of ρ is at least $1/\alpha C$ times the value of the $i'th$ coordinate of CR^*.*

For example,$(0.9918, 0.9918, 0.9918, 1.4853, 1.4902)$ is 0.01-approx. to $(1, 1, 1, 1.5, 1, 5)$

[4]For the lexicographical order between two vectors v and u we examine them ordered in increasing order, ζ and ξ, respectively. We say that $v > u$ if there is an index i, such that, $\zeta_i > \xi_i$ and $\zeta_j = \xi_j, 1 \leq j \leq i - 1$. Namely we find the longest equal prefixes of the two ordered vectors and define the order according to the first element which is not equal.

The centralized weighted max-min fair algorithms *OPT_WMMF* that uses an LP solver finds the optimal max-min fair rate vector in a variable-routing scenario. The centralized ϵ_*WMMF* and the distributed ϵ_*WMMFdist* algorithms find the maximal max-min fair rate vector in a variable-routing scenario. Due to the on-line nature of these algorithm, the found routing could not be guaranteed to be the global max-min fair vector.

B. Maximum Concurrent Multi-Commodity Flow Problem

The **Maximum Concurrent Multi-Commodity Flow** problem is stated as follows. Let G=(V,A) be a directed graph with nonnegative capacities $c(a), \forall a \in A$. If $a \notin A$ $c(a) = 0$. There are K commodities $C_1, \ldots, C_K$, each is specified by the triplet $C_i = (s_i, t_i, dem_i)$. The pair (s_i, t_i) are the source and the sink of commodity i, respectively, and dem_i is its rate demand. Each pair is distinct, but vertices may participate in different pairs. The objective is to maximize λ, s.t., for $i = 1, \ldots, K$ $\lambda \cdot dem_i$ units of the respective commodities can be simultaneously routed, subject to flow conservation and link capacity constraints. The objective λ is the equal maximal fraction of all demands. The following linear program *MC_MCF* primal is a path flow formulation that assigns the maximum commodity flow to P_i, the set of all paths between s_i and t_i that belong to the same commodity i, such that the assignment is restricted by the fairness criterion. PR's solution is composed of the assigned net flow, $f(P_{ij}) \forall P_{ij} \in P_i, i = 1, \ldots, K$, and the maximal fair fraction, λ.

MC_MCF **primal LP: Path Flow Formulation**

maximize λ

subject to

$$\forall a \in A, a \in P, \sum_{i=1}^{K} \sum_{P \in P_i} f(P) \leq c(a) \qquad (1)$$

$$\forall i, \sum_{P \in P_i} f(P) \geq \lambda \cdot dem_i \qquad (2)$$

$$\forall P \in P_{i=1\ldots K} f(P) \geq 0, \lambda \geq 0$$

This problem can be solved optimally in a polynomial number of steps. The size of this linear problem grows with the number of possible paths between any pair of nodes and can be exponentially large when the network is highly connected. It can be solved using the ellipsoid algorithm or by using other LP solver that solves an *arc flow formulation* of this problem. Note that the route assignment for some per-commodity bandwidth allocation is not unique.

The following is a description of the LP dual to *MC_MCF* problem. The $l(a)$ variable holds the link

length which is dual to each primal capacity constraint. The $z(i)$ variable holds the shortest path per each commodity and is dual to the demand portion constraint. The minimization problem can be stated as finding the minimum cost of shipping dem_k units from s_k to t_k where $l(a)$ is the price of shipping one unit along link a. Thus, the dual objective is to minimize the function $D(l) = \sum_{a \in A} c(a)l(a)$. Let $\alpha(l) = \sum_i dem_i \cdot dist_i(l)$ where $dist_i(l)$ is the shortest path length between the pair (s_i, t_i). Minimizing $D(l)$ is equivalent to computing the length $l(a)$ per each link which minimizes $D(l)/\alpha(l)$ such that the dual target β is equal to $\min_i D(l)/\alpha(l)$.

DUAL minimization LP

$$\text{minimize } D(l) = \sum_{a \in A} c(a)l(a)$$

subject to

$$\forall i = 1 \ldots K, \forall P \in P_i, \sum_{a \in P} l(a) \geq z(i) \quad (3)$$

$$\sum_{i=1}^{K} dem_i z(i) \geq 1 \quad (4)$$

$$\forall a \in A, l(a) \geq 0, \forall i = 1, \ldots, K, z(i) \geq 0$$

III. OPT_WMMF - Optimal Weighted Max-Min Fair Algorithm

OPT_WMMF is an off-line, centralized algorithm that calculates the global max-min fair vector using an LP solver. The commodity rate vector that is calculated in OPT_WMMF is optimal, i.e., 1-coordinate-wise competitive.

The OPT_WMMF algorithm (see Fig. 3) receives as input the list of commodities, Γ; the vector of demands, dem; and the graph, G. It initializes Γ_{UNSAT}, the list of unsaturated commodities (the commodities that can still increase their bandwidth assignment), to all the commodities; and Γ_{SAT}, the list of saturated commodities to null. It proceeds in iterations. In each iteration the algorithm lays new routes per all the commodities over G and increases the allocated bandwidth of the unsaturated commodities by solving a number of LPs, each is a reformulation of the MC_MCF problem.

There are a few goals to each iteration. The first goal is to maximize, λ, the equal share of all the unsaturated commodities in a fair manner and under the restriction of arc capacities. The second goal is to find a routing in G for all the bandwidth allocated, both in the previous iterations and the current one. At the end of the iteration, some of the commodities become saturated and are thus removed from Γ_{UNSAT} and added to Γ_{SAT}. This final decision, whether an unsaturated commodity becomes saturated at the end of an iteration, is the most difficult.

Since, at least, one new commodity becomes saturated at the end of each iteration the algorithm converges.

The following OPR LP is a reformulation of the MC_MCF problem (equations 1-2)

$$\text{maximize } \lambda$$

subject to

$$\forall a \in A, a \in P, \sum_{i=1}^{K} \sum_{P \in P_i} f(P) \leq c(a) \quad (5)$$

$$\forall i \in \Gamma_{unsat}, \sum_{P \in P_i} f(P) \geq \lambda \cdot dem_i \quad (6)$$

$$\forall i \in \Gamma_{sat}, \sum_{P \in P_i} f(P) \geq \lambda_{\phi_i}^{sat} \cdot dem_i \quad (7)$$

$$\forall P \in P_{i=1 \ldots K} f(P) \geq 0, \lambda \geq 0$$

Eq. 6 restricts the unsaturated commodities by λ, the equal share (weighted by the appropriate demand) that all the commodities can use. The objective of the OPR problem is to maximize this λ that appears only in Eq. 6. The allocated bandwidth to the saturated commodities was already assigned in the previous iterations. A saturated commodity i that became saturated in iteration ϕ_i and was assigned a bandwidth of $\lambda_{\phi_i}^{sat} \cdot dem_i$, where $\lambda_{\phi_i}^{sat}$ is the value of λ that was calculated in iteration ϕ_i. Eq. 7 preserves the already assigned bandwidth for the saturated commodities. Eq. 5 is the per-arc capacity constraint. By solving the OPR problem during iteration k, we find the additional possible equal share of bandwidth per each unsaturated commodity and find routing for all the commodities, unsaturated and saturated.

The OPR LP performs two tasks: maximizing the equal per-commodity allocation increase and routing re-assignment, such that previously allocated commodities will not steal bandwidth from the unsaturated commodities by occupying their more preferential paths. We will illustrate later the latter point. The OPR's solution is composed of the maximal fair fraction, λ, for the unsaturated commodities, the set of paths, $P_{ij} \in P_i$, $i = 1, \ldots, K$, and the assigned net flow per each path, $f(P_{ij})$.

Fig. 3 presents a pseudo-code of the OPT_WMMF algorithm. The algorithm iterates (line 5) each time with a reduced number of commodities (line 23). In each iteration it first solves the OPR LP (Eqs. 5-7). Lines 11-25 perform a two-phase test in order to identify which commodities become saturated in this iteration. In the first phase a simple connectivity test is performed where the residual graph can be calculated using the net flows, which were allocated in line 7[5].

---

[5]It is easier to calculate the residual graph when the arc flow formulation is used

$OPT\_WMMF(\Gamma, dem, G)$

1. /* Initialization stage */
2. $\Gamma_{UNSAT} = \Gamma$
3. $\Gamma_{SAT} = null$
4. $\phi = 0$ /* Iteration counter */

5. **while** $(\Gamma_{UNSAT} \neq null)$ **do** /*ITER*/
6. $\phi + +$
7.   **Perform** LP OPR
8.   **Using** G, Γ_{SAT}, Γ_{UNSAT}, and dem
9.   **Returns**: $\lambda_\phi, f(P), \forall_{P \in P_i}$
10.   /*Per Commodity Two-Phase Saturation Test */
11.   **for** commodity $k \in \Gamma_{UNSAT}$ **do**
12.   /* Phase I - connectivity test */
13.   **if** G has no connectivity for k **then**
14. $\Gamma_{tmpSAT} = \Gamma_{tmpSAT} \bigcup \{k\}$
15.   /* Phase II - saturation test */
16.   **for** commodity $j \in \Gamma_{tmpSAT}$ **do**
17.   **Perform** LP OPR
18.   **using** $\Gamma_{UNSAT} = j, \forall k \neq j : \lambda_{\phi_k}^{sat} = \lambda_\phi$
19. $\Gamma_{SAT} = \Gamma \setminus \{j\}$.
20.   **Returns**: $\lambda_{temp}, f(P), \forall_{P \in P_i}$
21.   **if** $\lambda_{temp} = \lambda_\phi$ **then**
22. $\Gamma_{SAT} = \Gamma_{SAT} \bigcup \{j\}$
23. $\Gamma_{UNSAT} = \Gamma_{UNSAT} \setminus \{j\}$
24. $\phi_j = \phi$
25. $\lambda_{\phi_j}^{sat} = \lambda_\phi$
26. /*end of while*/
27. **Returns** per commodity k: set of paths P_k and $f(P) \forall_{P \in P_k}$

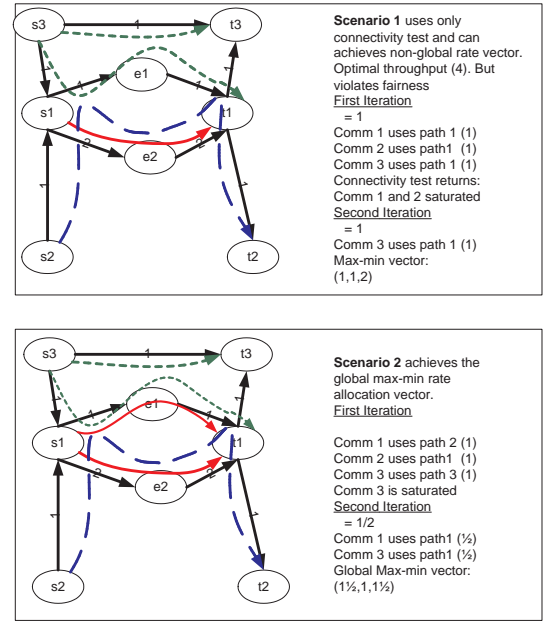Fig. 3. *OPT_WMMF* Optimal Weighted Max-Min Fair multi-commodity Algorithm

Fig. 4. A mistaken saturation example. Given three commodities: Commodity 1 (s1,t1) with the paths path1 (s1-e1-t1) and path2 (s1-e2-t1); Commodity 2 (s2,t2) with the paths path1 (s2-s1-e1-t1-t2) path2 (s2-s1-e2-t1-t2); and commodity 3 (s3,t3) with the paths path1 (s3-s1-e1-t1-t3), path2 (s3-s1-e2-t1-t3), and path3 (s3-t3). The different routing in iteration 1 of the two scenarios achieve the same throughput. However, in scenario 1 (upper part of the figure) the routing of commodity 3 during the 1st iteration leaves no room for adding flow to commodity 1 in the second iteration which makes it look 'saturated', and, indeed, the allocation vector of scenario 2 is fairer. Phase 2 of the saturation test forces helps to identify commodity 2 as unsaturated, and the new LP finds an optimal routing assignment as in scenario 2.

The reason for the two-phase saturation test is to cheaply identify candidates for saturation, using the criteria of the lack of connectivity in the residual graph, and then perform the costlier saturation test of phase two for these candidate commodities. Before explaining the phase two saturation tests, we direct the reader to Fig. 4 that illustrates a case where an unsaturated commodity can be mistakenly considered saturated when using only the cheaper connectivity test, which is the raison d'être for the second phase.

The second phase of the saturation test appears in lines 17-20 of Fig. 3. The OPR LP (line 17) is performed for each commodity that was suspected to be saturated by the connectivity test in order to find out whether it can increase its assignment considering a different routing. For this purpose we change the LP definition as follows. In Eq. 6 Γ_{unsat} holds only commodity j, whereas the other commodities are added to Γ_{sat} (Eq. 7), as if they are saturated and λ_ϕ^{sat} is set to their last calculated λ. If

the newly calculated λ for j is the same as before, then j is considered a saturated commodity (lines 22-25).

The *OPT_WMMF* algorithm provides us with a commodity rate vector ρ, a set of flow rate vectors f_k, and the rates of the paths $P_k^j \in P_k, k = 1, \ldots, K$ per commodity k, composing each ρ_i.

The following Lemmas and Theorem show that the two phase saturation test is true under any routing combination and that the obtained rate vector is the globally weighted max-min fair rate vector. The proofs of the Lemmas can be found in the technical report version of this paper.

Lemma 1 *Commodity k that was identified as* saturated *at the end of iteration ϕ can not increase its bandwidth allocation under any routing combination of the other commodities.*

Proof: Iteration ϕ starts with $\Gamma_{sat}^{\phi-1}$, the list of the commodities that were identified as *saturated* in any iteration ϕ' such that $\phi' < \phi$, and $\Gamma_{unsat}^{\phi-1}$, the commodities that can still grow. The LP OPR (line 7)

computes λ_ϕ for any commodity in $\Gamma_{unsat}^{\phi-1}$.

The first phase checks connectivity per each commodity. The commodities which can still be assigned bandwidth between its source and destination can still grow and remains in Γ_{unsat}^{ϕ}.

Assuming, by contradiction, a commodity l that was assigned λ_{temp} that is equal to λ_ϕ by the solution of OPR in line 17, and still can increase its bandwidth. The first operation of LP OPR (line 7) guarantees at least λ_ϕ per each commodity in $\Gamma_{unsat}^{\phi-1}$. By fixating this bandwidth allocation, the second operation of OPR, maximizes λ_{temp} under any routing combination of the other commodities. Any possible increase of commodity l is in contradiction to the optimality of the second OPR in line 17 operation. ∎

Lemma 2 *The commodity rate vector ρ provided by the OPT_WMMF is weighted max-min fair.*

Proof: We prove by induction on the iteration number n. The induction base holds for the first iteration since ρ^1, the allocated rate vector in the first iteration, equals to the solution of PR. PR found λ_1 to be the maximum portion can be assigned considering the demands and the capacities. It holds per all commodities i and j that $\rho_i^1/\rho_j^1 = dem_i/dem_j$. Assume now, that the induction hypothesis holds for iteration n. Thus, the rate vector solution is feasible and for each commodity i, ρ_i^n cannot be increased without decreasing any other ρ_j^n for some commodity j for which $\rho_i^n/\rho_j^n \geq dem_i/dem_k$.

We prove now the induction hypothesis holds for iteration $n+1$. According to lemma 1, the commodities that are in Γ_{UNSAT} can still grow and the others that are in Γ_{SAT} can not increase their bandwidth. The operation of OPR preserves the allocated bandwidth for all the commodities in Γ_{SAT}, such that the gained increase for any commodity $l \in \Gamma_{unsat}^{n+1}$ is calculated by OPR while keeping fairness and throughput, and not on the account of any commodity in Γ_{sat}^{n+1}. The saturation test guarantees the saturation of Γ_{sat}^{n+2}. ∎

Theorem 1 *The commodity rate vector ρ provided by the OPT_WMMF is optimal weighted max-min fair.*

Proof: Lemma 1 proves the correctness of the saturation test, which implies its maximum value under any routing scenario at its saturation iteration. This is true per each commodity ∎

IV. LOCALLY OPTIMAL WEIGHTED MAX-MIN FAIR ALGORITHMS

The *OPT_WMMF* algorithm presented in the previous section finds the globally max-min fair rate vector by looking for a new routing in each iteration. This re-assignment prevents us from using this algorithm in an on-line or distributed manner. However, due to practical reasons, a distributed version that finds a max-min fair rate allocation vector is important. Thus, we relax the requirement for a globally max-min fair vector and accept any max-min fair vector which is locally optimal.

This section presents an off-line centralized algorithm, *LOC_WMMF*, that finds the maximal max-min fair rate vector using an LP solver. Then, we present ϵ_*WMMF* (In subsection IV-A), an FPTAS approximation for *LOC_WMMF*, and a distributed version of this algorithm, ϵ_*WMMFdist* (In subsection IV-B). We show the practicality and the efficiency of ϵ_*WMMF* and ϵ_*WMMFdist* algorithms in subsection V.

LOC_WMMF is an iterative algorithm, that increases in each iteration the allocated net flow per commodity while keeping the weighted fairness criterion. It guarantees the max-min fairness since it is based on the optimality and scalability of the Maximum Concurrent Multi-Commodity Flow problem. It performs the *MC_MCF* LP (using Eqs. 1-2 and not the reformulated equations as in *OPT_WMMF*) on the residual graph of G, and preserve the routing that was calculated in previous iterations. The algorithm iterates each time with a reduced number of commodities until the total net flow per a commodity is assigned. One must be careful in such algorithms since the inability to reroute may result in a solution which is away from the global as illustrated in Fig. 2 where two routing layouts, which can be found by the *LOC_WMMF* algorithm are presented. Only scenario two produces the global weighted max-min fair rate vector. Thus, ϵ_*WMMF* and ϵ_*WMMFdist*, the approximation algorithms for *LOC_WMMF*, route a small amount of flow (with respect to the link weight in the dual problem) in each iteration. More details about *LOC_WMMF* appear in [21].

A. Weighted Max-Min Fair Centralized Approximation Algorithm

The ϵ_*WMMF* algorithm (see Fig. 5) is an ε-approximation of the *LOC_WMMF* algorithm. It uses the variable-size increment technique (which is close in spirit to the primal-dual techniques) instead of the LP solver used in *LOC_WMMF*. The routing is controlled by the tight relationship between the length of the selected path and its load; a fact that spreads the routes and increases network utilization (thus avoiding cases like in Fig. 2 Scenario one).

An FPTAS approximation for the *MC_MCF* problem that uses a variable-size increment technique is developed and presented in [18], [19], [20]. They treated the

$\epsilon\_WMMF(\Gamma, dem, G, \epsilon)$
1. /* Initialization stage */
2. $\forall a \in A, l(a) = \delta/c(a)$
3. **while** $(\Gamma \neq NULL)$ **do** /* STAGE*/
4. $stageCnt++, phaseCnt = 1$
5. $lastDL = 0; newDL = D(l)$
6. **while** $(newDL - lastDL < 1)$ **do** /* PHASE */
7.   **for** $(i = 1 \ to \ |S|)$ **do** /*ITER: S group of diff srcs*/
8.   Let Γ_i group of commodities start from source i
9.   Build shortest path tree, $P^* = \{P_C^* | C \in \Gamma_i\}$
10. $\forall C \in \Gamma_i, tmpdem(C) = dem_C$
11.   **while** newDL-lastDL < 1 and
12. $\exists C \in \Gamma_i, tmpdem(C) > 0$ **do** /*STEP*/
13.   **for** $C_k \in \Gamma_i$
14.   /* Connectivity and Saturation test */
15.   **If** $\forall a \in P_{C_k}^*, l(a) \geq 1/c(a), l$ **then**
16. $\Gamma_i = \Gamma_i \setminus \{C_k\}$
17. $\Gamma = \Gamma \setminus \{C_k\}$
18.   **else** /*Update Curr Path*/
19. $c = \min_{a \in P_{C_k}^*} c(a)$
20. $f_{C_k} = \min(tmpdem, c)$
21. $f(P_{C_k}^*) = f(P_{C_k}^*) + f_{C_k}$
22. $tmpdem(C_k) = tmpdem(C_k) - f_{C_k}$
23.   **end for**
24. $\forall a \in P_{C_k}^*, l(a) = l(a)(1 + \epsilon \cdot \sum_{C_k : a \in P_{C_k}^*} \frac{f_{C_k}}{c(a)})$
25. $newDL = D(l)$
26.   **end while** /* end of step */
27.   **end for** /* end of iteration */
28. $phaseCnt++;$
29. **end while** /* end of phase */
30. $lastDL = newDL$
31. **end while** /* end of stage */
32. $\forall k = 1 \ldots K, \forall P \in P_k, f(P) = \frac{f(P)}{\log_{1+\epsilon} \frac{1+K\epsilon}{\delta}}$
33. $\forall k = 1 \ldots K, \forall P \in P_k, f(P_k) = \sum f(P)$
34. **Returns** per commodity k:
35.   set of paths P_k and flows $f(P_k)$

Fig. 5. ϵ_WMMF Approximation Optimal Weighted Max-Min Fair multi-commodity Algorithm

MC_MCF- equations 1 and 2 - formulation as the primal problem and equations 3and 4 as the dual problem. Our algorithm extends their technique such that it iterates on the dual variables until all the shortest paths are saturated. While the referred works did not deal with the saturation issue, we suggest a saturation test (which also serves as a connectivity test) that enables us to stay in the dual problem. Another advantage of sticking with the dual problem is the reflection of the fairness among the commodities, which is our primal objective, using the dual objectives and variables. In addition to the fairness, we show that these variables can be used to determine the saturation of a path.

The algorithm receives as input the list of commodities Γ, the vector of demands dem, the graph G and ϵ, the maximum allowed approximation error. It starts by assigning the length of each link $l(a)$ to be $\delta/c(a)$, where δ is a pre-computed value chosen to achieve the desired approximation value. The algorithm alternates between primal flow variables and dual length variables to fulfill the capacity-length constraint (primal Eq. 1 and dual Eq. 3). It proceeds in stages (see line 3 in Fig. 5). In each stage the algorithm solves the *MC_MCF* problem (using an approximation algorithm taken from [18], [19], [20]) and finds the primal-dual (λ and $D(l)$) solution. Part of the commodities become saturated during each stage and should be omitted in the following stages. The saturation test is an important contribution of our algorithm that promises the reduction in the number of the participating commodities at each stage and thus the convergence of the algorithm.

The stage proceeds in phases (line 6). Each phase is composed of $|S|$ iterations, where S is the group of all the sources (some commodities can have the same source s_i). Iteration i of phase j considers the commodities $C_q, q = 1 \ldots r$ starting from the same source S_i (see line 7) and routes $dem(C_q)$ units in a number of steps.

Each step (see line 11) calculates the shortest path tree starting from the source, using the last calculated length variables $l(a)$. It iterates over the q commodities and in each step either saturates the current shortest path per commodity C_q or allocates the remained $dem(C_q)$. For every $c(a)$ units of flow sent over the link a, its link length variable $l(a)$ is updated by a factor of at most $1 + K\epsilon$ (line 24). The entire stage ends as soon as $D(l) \geq 1$ according to the dual constraint Eq. 4 and produces a vector of lengths, l. The corresponding per commodity net flow vector $f(P_k), k = 1 \ldots K$ is infeasible for the primal LP, and needs to be scaled down. For this purpose, we note that as long as $D(l) < 1$, the length of each link can not exceed $1/c(a)$, which implies that the number of times the flow is increased over this link (during a stage period) is $\log_{1+\epsilon}(\frac{1+K\epsilon}{\delta})$ times its real flow. By scaling down this flow by a factor of $\log_{1+\epsilon} \frac{1+K\epsilon}{\delta}$, a feasible flow will be achieved. The scaling is done after the termination of all the phases (line 31). Since the scaling factor is known in advance, the scaling can be done at any point within the step and thus the feasible value of the flow can be followed.

Iterating over $|S|$ is more efficient than iterating over the commodities since the entire shortest path tree is calculated at once instead of one shortest path calculation at a time. We will use this improvement [20] in the distributed implementation. The connectivity test per each

commodity is also done at this point by checking the unsaturated shortest path per the participating commodities. Only the commodities that pass the connectivity test participate in this stage. Note that this check is done while building the shortest path tree and thus no extra running time is needed for this test.

The primal-dual solutions are found when the function $D(l)$ is larger than 1. Since, each stage is an activation of the primal-dual alternation, during stage i we achieve a primal-dual solution β_i and z_i which are found when the function $D(l)$ is larger than 1. In order to saturate the network, we continue to increase the length variables, $l(a)$, but the termination condition $(D(l) > 1)$ should consider only the additional length for the last stage. Thus, the $l(a)$ variables hold the accumulative length values and are used for the shortest path calculations. However, for the stage termination condition we consider only the incremental values, namely, $newDL - lastDL$ (lines 6 and 11), where $lastDL$ is the $D(l)$ value at the beginning of the stage and $newDL$ is the current value of $D(l)$ (line 25). At each stage, at least one commodity is saturated and removed from the list Γ since, at least, one link value is increased by a factor of $(1+\epsilon)/c(a)$. This ensures the algorithm convergence. **Algorithm correctness and complexity** Past analysis [18], [19], [20] showed the correctness of the *MC_MCF* approximation algorithm and proved the dual-primal solution ratio, β/λ, to be less than $1 + \xi$ for any $\xi > 0$. For the *MC_MCF* problem they proved the following theorem.

Theorem 2 *(Lemma 3.2 and Theorem 3.1 in [20]) There is an algorithm that computes a $(1-\epsilon)^{-3}$-approximation to the MC_MCF in time $O(\epsilon^{-2}m^2 \log m)$ where m is the number of edges.*

The solution of ϵ_*WMMF* algorithm is an approximation to some weighted max-min rate vector, not necessarily the global one. Next, we will prove the feasibility, convergence, and the runtime of this algorithm. Theorem 3 proves the approximation bound and the runtime complexity of ϵ_*WMMF*.

Theorem 3 *The ϵ_WMMF algorithm computes a $(1-\epsilon)^{-3}$-approximation to some weighted max-min fair flow in time $O(\epsilon^{-2}Km^2 \log m)$, where m is the number of edges.*

Proof: The analysis and proof in [18], [19], [20] hold for one stage of the ϵ_*WMMF* algorithm. The analysis follows the ones in the above mentioned references, but here we examine the number of phases in all the stages. Let $D(l_i) = \sum l(a) \cdot c(a)$ and $\alpha(l_i) =$ $\sum_q dem(q) \cdot dist_q(l_i)$ where $dist_q(l_i)$ is the shortest path length between the pair (s_q, t_q) for the length assignment l_i at phase i. The $D(l)$ function is increased at each phase as follows:

$$D(l_i) \leq D(l_{i-1}) + \epsilon\alpha(l_i) \tag{8}$$

Considering the dual result $\beta_k = \min_i D(l_i)/\alpha(l_i)$ during stage k, $\beta = \sum_k \beta_k$ and substituting these values in Eq. 8, the following holds

$$D(l_i) \leq \frac{D(l_{i-1})}{1 - \epsilon/\beta} \tag{9}$$

Since $D(l_{t_s}) \geq 1$ holds for any stage, where t_s is the total number of phases per this stage, we can assume that $D(l_t) \geq K$, where t is the total number of phases over all the stages and K is the number of commodities.

Using $D(l_0) = m\delta$, $\beta \geq 1$ and $D(l_t) \geq K$, the following holds

$$K \leq D(l_t) \leq \frac{m\delta}{1 - \epsilon} e^{\frac{\epsilon(t-1)}{\beta(1-\epsilon)}} \tag{10}$$

and a simple Algebraic manipulation yields

$$\beta \leq \frac{\epsilon(t-1)}{(1-\epsilon)\ln\frac{K\cdot(1-\epsilon)}{m\delta}} \tag{11}$$

Using the claim from [20] for each of the stages, summing up, and substituting in Eq. 11 we get

$$\beta/z < \frac{\epsilon}{(1-\epsilon)\,\ln(1+\epsilon)\cdot K} \cdot \frac{\ln\frac{1+K\epsilon}{\delta}}{\ln\frac{K(1-\epsilon)}{m\delta}} \tag{12}$$

By setting δ to be

$$\delta = \frac{1}{(1+K\epsilon)^{\frac{(1-\epsilon)}{(K-(1-\epsilon))}}} \cdot \left(\frac{K(1-\epsilon)}{m}\right)^{1+\frac{1-\epsilon}{K-(1-\epsilon)}} \tag{13}$$

The β/λ ratio, which is the primal dual ratio calculated by the ϵ_*WMMF* algorithm, becomes less than $(1-\epsilon)^{-3}$ and any ϵ can be selected.

It now remains to show that the resulted rate vector is indeed max-min fair. This can be done by noticing the analogy between the operation of *LOC_WMMF* and ϵ_*WMMF*, and the proof of the correctness of *LOC_WMMF* in [21]. The proof can be found in the full version of this paper [22]. ∎

B. Weighted Max-Min Fair Distributed Approximation Algorithm

We assume that each source router is familiar with the network topology, link capacities, the commodities emanating from it, and the ids of all other sources. In addition, the sources are required to synchronize at the end of each phase as explained later. During a phase each source independently performs its procedure, iterating over the steps. The distributed implementation of our

```
    ε_WMMFdist-source(Γ_i, dem, G, ε)
 1. /* Initialization stage */
 2. ∀a ∈ A, l(a) = δ/c(a)
 3. while (Γ_i ≠ NULL) do  /* STAGE*/
 4.   stageCnt + +, phaseCnt = 1
 5.   lastDL = 0; newDL = D(l)
 6.   while (newDL − lastDL < 1) do /* PHASE */
 7.     Let Γ_i group of commodities start from source i
 8.     Build shortest path tree, P* = {P*_C | C ∈ Γ_i}
 9.     ∀C ∈ Γ_i, tmpdem(C) = dem_C
10.     while newDL-lastDL < 1 and
11.        ∃C ∈ Γ_i, tmpdem(C) > 0  do /*STEP*/
12.        parallel for C_k ∈ Γ_i
13.          /* Connectivity and Saturation test */
14.          If ∀a ∈ P*_{C_k}, l(a) ≥ 1/c(a), l then
15.             Γ_i = Γ_i \ {C_k}
16.             Γ = Γ \ {C_k}
17.          else /*Update Curr Path*/
18.             SndSRCAlc(tmpdem(C_k), P*_{C_k})
19.             WaitDestMsg(P*_{C_k}, f(P*_{C_k}), l)
20.             tmpdem(C_k) = tmpdem(C_k) − f(P*_{C_k})
21.             newDL(l) = Σ_{a∈A} c(a)l(a)
22.        end parallel for
23.      end while  /* end of step */
24.      SndAl2AlSRCMsg(i, l, phaseCnt, stageCnt)
25.      GtAlSRCSYNMsg(l, phaseCnt, stageCnt)
26.      newDL(l) = Σ_{a∈A} c(a)l(a)
27.      phaseCnt + +;
28.    end while  /* end of phase */
29.    lastDL = newDL;
30. end while  /* end of stage */
31. ∀k = 1...K, ∀P ∈ P_k, f(P) = f(P) / log_{1+ε} (1+Kε)/δ
32. ∀k = 1...K, ∀P ∈ P_k, f(P_k) = Σ f(P)
33. Returns per commodity k:
34.    set of paths P_k and flows f(P_k)
```

Fig. 6. ε_WMMFdistDistributed Approximation Optimal Weighted Max-Min Fair multi-commodity Algorithm for source node i

algorithm is shown in Fig. 6 for the source node; the code for an intermediate node or a destination node is omitted due to space limitations, but explained below. In each step, the source sends an "Allocation Request" message (line 18) over the pre-calculated shortest path tree (line 8). This message traverses all intermediate nodes towards the destinations and collects the information about the bottleneck link along each shortest path. The destination node, upon source message arrival, sends a "Destination Acknowledge" message with the capacity of bottleneck link. Each intermediate router receives the "Destination Acknowledge" message, updates the length and the flow over its outgoing arc variables (using the same formulae as in lines 21 and 24 in Fig. 5), and

forwards it towards the source. The source receives the message (line 19) and updates per-arc length information and per-path bandwidth allocation amount. Each source node, at the end of a phase, synchronizes with the other sources by exchanging the length information (lines 24 and 25). By having all source nodes registered to a multicast group, one can simplify the synchronization process. This distributed algorithm proceeds until the network is saturated and each commodity obtain its fair share.

In case a commodity is dropped from the demand list we need not run the algorithm from scratch. A commodity can be dropped once the algorithm terminated or even at any intermediate synchronization state. This incremental change requires adding another message type that will be sent from the source node to the destination along the dropped commodity paths. Each intermediate link can reduce its length and divide it by a factor of $1 + \epsilon \frac{f_k}{c(a)}$ for this specific flow. Since the algorithm is performed over the dual variables, this will be enough to adjust the state of network bandwidth allocation. After this adjustment, the network becomes unsaturated for at least some of the sources and the algorithm continues until saturation is achieved. The incremental algorithm for the case of adding a commodity is left for future research.

V. ALGORITHM IMPLEMENTATION AND PERFORMANCE RESULTS

We implemented the *OPT_WMMF* and ε_WMMF algorithms using MATLAB in order to study their accuracy and performance. Specifically we show that the ε_WMMF algorithm solution is close to the one achieved by the *OPT_WMMF* optimal algorithm, while its running time is significantly shorter.

A. Algorithm Implementation

In this section we present the simple example of Fig. 7 to illustrate the way the algorithm iterates. The capacity of each link is 1. There are four commodities, each with 1 unit of demand. All the links and paths are unidirectional. Commodities 2 and 4 have one path and its path ID is 1. Commodity 1 and 3 have 2 paths with IDs 1 and 2.

Table II presents the two stages of the algorithm operation for $\epsilon = 0.2$. We can see that in the first stage all the commodities receive an equal portion of their demands. Link 2 is the bottleneck link of their paths and its length after this stage becomes $1.1451 > 1/c(2) = 1$. It means that this link is saturated. We can verify it by observing its flow which is $0.3258 + 0.3438 + 0.3258 =$
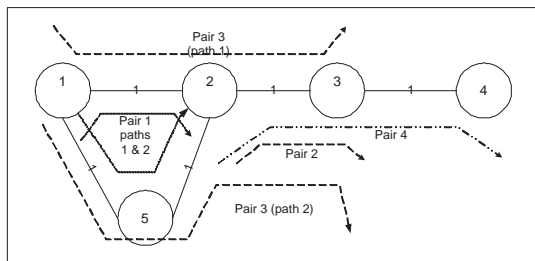
Fig. 7.   Algorithm Iteration Example

comm. ID	path ID	infeasible flow	feasible flow	per comm. flow	path length
stage 1					
1	1	2	0.0362	0.3438	
1	2	17	0.3077		0.0019
2	1	18	0.3258	0.3258	0.6627
3	1	19	0.3438	0.3438	0.9562
3	2	0	0.0		0.9562
4	1	18	0.3258	0.3258	0.7963
$l = \{0.552\ 1.145\ 0.001\ 0.266\ 0.266\}$, $\lambda = 0.326$, lastDL = 1.151					
stage 2					
1	1	32	0.5791	1.4297	0.4602
1	2	47	0.8506		∞
2	1	18	0.3258	0.3258	∞
3	1	19	0.3438	0.3438	∞
3	2	0	0.0		∞
4	1	18	0.3258	0.3258	∞
$l = \{1.145\ 1.145\ 0.001\ 0.552\ 0.552\}$, $\lambda = 0.326$, lastDL = 2.231					

TABLE II

ϵ_WMMF EXECUTION USING FIG. 7 AND $\epsilon = 0.2$

Run 1					
source-dest	2-5	13-12	5-12	1-4	11-1
OPT_WMMF	0.5	0.5	1	2	2
ϵ_WMMF	0.4959	0.4959	0.9918	1.9059	1.9059
Run 2					
source-dest	1-2	3-2	1-10	2-8	6-13
OPT_WMMF	1	1	1	1.5	1.5
ϵ_WMMF	0.9918	0.9918	0.9918	1.4853	1.4902
Run 3					
source-dest	6-5	9-5	11-7	2-9	11-1
OPT_WMMF	0.5	0.5	1	2	2
ϵ_WMMF	0.4959	0.4959	0.9918	1.8816	1.9837

TABLE III

COMPARISON EXAMPLES OF *OPT_WMMF* AND ϵ_WMMF RATE ALLOCATIONS FOR THE TOPOLOGY IN FIG. 8.

US, which publish its network structure. This topology has many paths between most of it source-destination pairs. Such a topology is prone to problematic routing as we describe in Figures 4 and 2. We randomly selected a number of source-destination commodities for this topology, and for each random selection we compared the optimal rate vector which was found by *OPT_WMMF* algorithm with the rate vector which was found by the ϵ_WMMF algorithm. In the tens of comparisons we have made, each commodity in the ϵ_WMMF algorithm was allocated a rate which was, at most, ε below the optimal rate. Table III shows a few examples.

0.9954. The calculated λ for this stage is 0.3258 and the stage terminates when $D(l) = 1.1510$. Path 2 of commodity 3 does not get any flow due to the saturation of link 2. The other path of commodity 3 gets its fair share. At the second stage the algorithm discovers that commodities 2, 3, and 4 are saturated and delete them from Γ. In the following stage, the algorithm iterates for commodity 1 between its two shortest paths until the saturation of both. The final max-min vector rate for $\epsilon = 0.2$ and commodities 1 (path 1 and 2), 2, 3 (path 1 and 2) and 4 is $\{1.6469, 0.3258, 0.3438, 0.3258\}$. The final max-min rate vector for $\epsilon = 0.1$ for commodity 1 (path 1 and 2) commodities 2, 3 (path 1 and 2) and 4 is $\{1.6585, 0.3317, 0.3317, 0.3317\}$. Note that when the ϵ decreases the values are approaching the optimal weighted max-min vector (5/3,1/3,1/3,1/3).

B. Algorithm Result Comparisons

In order to demonstrate the practicality of the ϵ_WMMF algorithm, we performed extensive simulations. Fig. 8 describes the topology of the Broadwing network, an ISP with a nation-wide presence in the
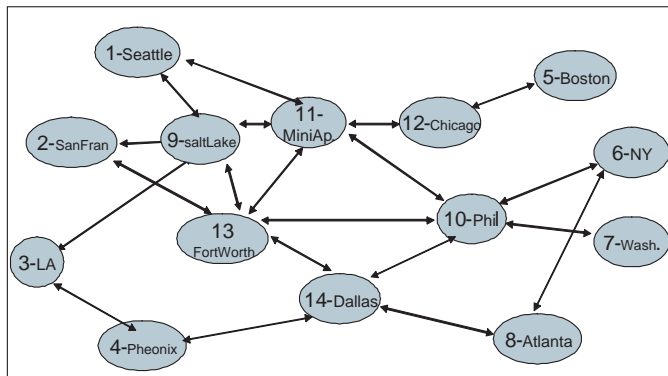

Fig. 8.   Topology map of Broadwing IP Routers connections

VI. CONCLUDING REMARKS

We presented, for the first time, an off-line centralized algorithm that finds the global max-min fair rate vector by using an LP formulation and solver. In addition we found a distributed, efficient and fast approximation for a traffic engineering algorithm for routing demands in a network in a way that maximizes the flows and maintains fairness. Our algorithm, which employs the

weighted max-min fairness criterion strikes the right balance between network utilization and fairness and thus serves the current needs for quality of service.

REFERENCES

[1] B. Shmoys, "Cut problems and their application to divide-and-conquer," in *Approximation Algorithms for NP-Hard Problems*, D. S. Hochbaum, Ed. PWS Publishing Company, 1997, ch. 5.

[2] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1992.

[3] Y. Afek, Y. Mansour, and Z. Ostfeld, "Phantom: a simple and effective flow control scheme," *Computer Networks*, vol. 32, no. 3, pp. 277–305, 2000.

[4] B. Awerbuch and Y. Shavitt, "Converging to approximated max-min flow fairness in logarithmic time," in *INFOCOM (2)*, 1998, pp. 1350–1357.

[5] Y. Bartal, M. Farach-Colton, S. Yooseph, and L. Zhang, "Fast, fair, and frugal bandwidth allocation in ATM networks," *Algorithmica (special issue on Internet Algorithms)*, vol. 33, no. 3, pp. 272–286, 2002.

[6] Y. Bartal, J. Byers, and D. Raz, "Global optimization using local information with applications to flow control," in *the 38th Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, 1997.

[7] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Operational Research Society*, vol. 49, pp. 237–252, 1998.

[8] J. Mo and J. Warland, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, 2000.

[9] S. Chen and K. Nahrstedt, "Maxmin fair routing in connection-oriented networks," in *Proceedings of Euro-Parallel and Distributed Systems Conference (Euro-PDS '98)*, 1998, pp. 163–168.

[10] J. M. Kleinberg, Y. Rabani, and E. Tardos, "Fairness in routing and load balancing," in *IEEE Symposium on Foundations of Computer Science*.

[11] N. Megiddo, "Optimal flows in networks with sources and sinks," *Math Programming 7*, 1974.

[12] A. Goel, A. Meyerson, and S. Plotkin, "Combining fairness with throughput: online routing with multiple objectives," in *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000, pp. 670–679.

[13] N. Buchbinder and J. Naor, "Fair online load balancing," in *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*. New York, NY, USA: ACM Press, 2006, pp. 291–298.

[14] Y. T. Hou, Y. Shi, and H. D. Sherali, "Rate allocation in wireless sensor networks with network lifetime requirement," in *MobiHoc '04: Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*. New York, NY, USA: ACM Press, 2004, pp. 67–77.

[15] V. V. Vazirani, *Approximation Algorithms*. Springer-Verlag, 2001.

[16] F. Shahroki and D. W. Matula, "The maximum concurrent flow problem," *Journal of the ACM*, vol. 37, pp. 318–334, 1990.

[17] N. Young, "Randomized rounding without solving the linear program," in *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, 1995, pp. 170–178.

[18] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," in *IEEE Symposium on Foundations of Computer Science*, 1998, pp. 300–309.

[19] L. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM J. Discrete Math*, vol. 13, no. 4, pp. 505–520, 2000.

[20] G. Karakostas, "Faster approximation schemes for fractional multicommodity flow problems," in *ACM SODA*, 2002, pp. 166–173.

[21] M. Allalouf and Y. Shavitt, "Maximum flow routing with weighted max-min fairness," in *First International Workshop on QoS Routing (WQoSR 2004)*, Barcelona, Spain, Oct. 2004.

[22] ——, "Centralized and distributed approximation algorithms for routing and weighted max-min fair bandwidth allocation," in *IEEE Workshop on High Performance Switching and Routing (HPSR'05)*, Hong Kong, May 2005.

PLACE PHOTO HERE

Miriam Allalouf received her B.Sc and M.Sc. degrees in computer science, from the Hebrew University, Jerusalem, Israel, in 1984 and 1987 respectively. Since 1985, she has several developing, designing and leading positions in the industry, mainly in networking and database areas. During the year of 96-97, she held a visiting position in the University of Massachusetts at Amherst, Dept. of Electrical and Computer Engineering, working on fault tolerance in real-time and parallel systems. During 97-2002 she participated in projects in the access networks field: as a traffic management group leader. She is currently terminating the Ph.D. degree in the School of Electrical Engineering at Tel-Aviv University and working at IBM Haifa Research Labs. Her current research interests include QoS subjects as traffic engineering & QoS Routing, and Bandwidth measurements analysis.

PLACE PHOTO HERE

Yuval Shavitt (S88M97SM00) received the B.Sc. degree in computer engineering (cum laude), the M.Sc. degree in electrical engineering, and the D.Sc. degree from the Technion, Israel Institute of Technology, Haifa, in 1986, 1992, and 1996, respectively. After graduation, he spent a year as a Postdoctoral Fellow at the Department of Computer Science at Johns Hopkins University, Baltimore, MD. Between 1997 and 2001, he was a Member of Technical Stuff at the Networking Research Laboratory at Bell Labs., Lucent Technologies, Holmdel, NJ. Starting October 2000, he is a Faculty Member in the School of Electrical Engineering at Tel-Aviv University Tel-Aviv, Israel. Dr. Shavitt served as a TPC member for INFOCOM 2000-2003 and 2005, IWQoS 2001 and 2002, ICNP 2001, IWAN 2002-2005, tridentcom 2005–2006, and more, and on the executive committee of INFOCOM 2000, 2002, and 2003. He was an Editor of Computer Networks 2003-2004, and served as a Guest Editor for JSAC and JWWW. His recent research focuses on Internet measurement, mapping, and characterization and on QoS in networks.