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

Miriam Allalouf
School of Electrical Engineering
Tel Aviv University
Email: miriama@eng.tau.ac.il

Yuval Shavitt
School of Electrical Engineering
Tel Aviv University
Email: shavitt@eng.tau.ac.il

Abstract

Given a set of demands between pairs of nodes, we examine the Traffic Engineering problem of maximal flow routing and fair bandwidth allocation where flows can be split to multiple paths (e.g., MPLS tunnels). In the past we presented a polynomial solution for this problem but its complexity makes it hard to implement for large problem sizes. Thus, this paper presents a fully polynomial epsilon-approximation (FPTAS) algorithm for the max-min fair allocation problem which is based on a primal-dual alternation technique. 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. We implemented the centralized algorithm 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 fairness will be preserved.

We consider as input a network topology and directional links 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, client ISP), ATM VPs, or MPLS tunnels, and will typically be expressed by average or maximum required rate. Thus, traffic between 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 keeping a fair sharing of the allocated bandwidth, to lay the set of paths to be used between each pair in the network, and to allocate them bandwidth in a maximal way. The fairness criterion is defined by the weighted max-min fairness.

One way to maximize the network flow is to formulate the problem as a maximum multi-commodity flow (MCF) 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.

The maximum concurrent multi-commodity flow (MCMCF) problem introduces fairness to the maximum flow problem. In this MCF LP formulation, we are given a set of K demands dem_i, one per each commodity pair (s_i, t_i) and require to satisfy the maximum equal fraction z of all demands and seeks a routing that maximizes network flow. However, the achieved solution under-utilizes the network, sometimes saturating only a small fraction of it.

The max-min fair allocation strikes a balance between fairness and the need to fully utilize the network. An allocation of bandwidths, or rates, to connection 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 any connections which have larger rates. The Max-min fairness criterion bandwidth allocation was mostly defined in the context of a single fixed path per session, where a session is defined by a pair of terminals.

This work focuses on an extended version of the max-min fair allocation where the flow between two terminals may be split among several paths. Furthermore, the solution we seek needs to find the set of the paths that achieve such maximal fair allocation chosen out of all possible paths. In addition, we use the weighted max-min fair version of the formulation to account for the demands.

The WMCM (Weighted Max-min fair Concurrent MCF) algorithm, developed in our previous work [1] finds the extended max-min fair rate vector in a polynomial number of steps. It solves iteratively the maximum concurrent LP until network saturation is achieved where each iteration performs the MCMCF LP over the residual capacity of the network with the commodities whose net flow can still be increased.

However, while the WMCM algorithm can be calculated in polynomial time, its running time depends on the LP solver in use, which may make it impractical for large inputs.

In this paper, we present a centralized and a distributed FPTAS approximation algorithms, called WMCMApprox and WMCMApproxDist, for calculating the rate vector of the weighted max-min fair problem. The centralized algorithm provides a faster way to solve the more complex version (as presented above) of the max-min fair allocation. As before we embed the MCMCF solution into the process of finding the rate vector of the max-min fairness flows. However, here the algorithm is different as it runs over the dual problem to the MCMCF and enables a more efficient centralized algorithm

and consequently, the distributed algorithm[1].

Our WMCMApprox algorithm embeds and extends the variable-size increments techniques (which appear in Garg and Könemann [2] and Fleischer [3]) to achieve a new solution to the max-min fair. The original form of 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 links length assignment.

Finally, our novel distributed algorithm, WMCMApprox-Dist, 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 rest of the paper is orgenized as follows. Section II presents the related works. Section III states definitions and explains the max-min fairness criteria in our context; it also describes the primal MCMCF problem and its dual problem. Section IV describes our new algorithms, the centralized algorithm and its implementation results using simple example, and the distributed algorithm. Section V summarizes the paper.

II. Related Work

The Max-min fairness bandwidth allocation 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 routing is given appears in [4].

Many other distributed algorithms deal with dynamic adjustments of flow rates to maintain max-min fairness when single routes are given [5], [6], [7]. The above algorithms differ by the assumptions on the allowed signaling, and available data. Bartal *et al.* [8] find the total maximum flow allocation in a network for given routes using distributed computations as the input to the global MCF LP problem.

The max-min fair problem with an unknown set of routes was rarely discussed. Kleinberg *et al.* [9] provide 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. They provide a maximum unsplittable flow allocation for single source commodities. Megido [10] addressed this problem for a single commodity maximum flow where the fairness achived among multiple sources and multiple sinks flows.

Chen and Nahrstedt [11] provide 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.

Maximum Concurrent MCF problem. Most of the studies that combined the LP formulation for the traffic engineering design chose an MCF formulation that considers the demands but they do not discuss the max-min fairness in conjunction with maximum throughput as the WMCM algorithm does[2]. A few directions for building approximation algorithms for the MCF problem were suggested in the past. Young [15] 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 [2] 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. Fleischer [3] and Karakostas [16] improved the Maximum multicommodity 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 algorithm extends these techniques to be used for the max-min fair allocation algorithms.

III. Definitions and Model

A. Max-Min fairness

To clarify the difference between the different fairness criteria and algorithms, consider the example in Figure 1, which depicts a line network with four nodes 1,2,3, and 4 and 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 bandwidth allocation is considered. The maximum MCF 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 [4] vector in this case is (1/3,2/3,1/3,1/3) which achieves a flow of 5/3. A weighted max-min fair algorithm as the WMCM and WMCMapprox algorithms treats each flow in this example as belong to a different commodity (with different source-destination pair). It will result, in case of equal weights (or demands) for all commodities (1/3,2/3,1/3,1/3), the same as the max-min vector given above. In case pair 1 is given a weight double than the rest of the nodes, the concurrent MCF problem will allocate it double the bandwidth allocated for the flows in its bottleneck link (link(2,3)) and the result weighted max-min vector is (1/2,1/2,1/4,1/4).

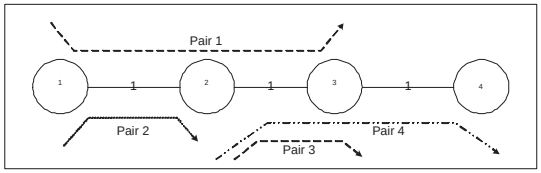

Fig. 1. Example of flows assignment

The classical max-min fair definition is stated for the case where each flow takes a single path [4], the following

[1]An approximation scheme is a family of algorithms 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.

[2]Relevant mathematical and algorithmic background on maximum concurrent MCF problem and its complexity can be found in [12], [13], [14].

definitions (See detailed description in [1]) extend it to the case where a flow may be split among several paths.

Definition 1 *The Commodity Rate Vector, cr, is a vector whose elements are the rates which were assigned to the commodities.*

From this definition we can write $\sum_{P_{ij} \in P_i} f_{ij} = cr_i$ where P_i is the set of all the paths that are assigned to commodity i. P_{ij} and f_{ij} are the $j-$th path and net flow of commodity i. The weighted max-min fair algorithm finds a commodity rate vector cr^* and a flow rate vector f_i per each commodity rate cr_i^*.

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

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

B. Maximum Concurrent Problem

The **Maximum Concurrent 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 different commodities: $C_1, \ldots, C_K$ where commodity i 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 z so all the $i = 1, \ldots, K$, $z \times dem_i$ units of the respective commodities can be routed simultaneously, subject to flow conservation and link capacity constraints. The objective z is the equal maximum fraction of all demands. The path flow formulation of the following linear program PR 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, restricted by the fairness criterion. The assigned net flow per arc a is the sum of the net flows of the paths passing this arc. 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, z.

PRIMAL LP PR: Path Flow Formulation

maximize z

subject to

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

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

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

This problem can be solved optimally in a polynomial number of steps; further discussion can be found in [1].

The following is a description of the LP that is dual to the Maximum Concurrent flow 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

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

subject to

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

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

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

IV. Weighted Max-min Fair Algorithms

A. Weighted Max-min Centralized Approximation Algorithm

The contributions of this section is a fast centralized approximation algorithms (FPTAS) for the WMCM which we term WMCMApprox. WMCMApprox (see Figure 2) is based on our optimal algorithm ideas but uses completely different techniques. It embeds previous approximation algorithm suggested for the maximum concurrent problem. Specifically we use the variable-size increment technique (which is close in spirit to the primal-dual techniques) and iterate on the dual variables until all the shortest paths are saturated. In the WMCM algorithm we suggested to check the residual graph after each maximum concurrent stage (as done for single path max-min flow algorithms [4]). Here we do not need to check this condition in the primal problem, and instead suggest a new saturation test (which also serves as a connectivity test) that enables us to stay in the dual problem. This provides us a faster runtime with an easy proof of the approximation ratio. 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 WMCMapprox approximation algorithm receives as input the list of commodities $KCOMM$, 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. 2). In each stage the algorithm solves the maximum concurrent problem (using approximation algorithm taken from [2], [3], [16]) and finds the dual-primal (z and $D(l)$) solution. Part of the commodities become saturated during each stage and should be omitted in the following stages. This is an important contribution

WMCMApprox($KCOMM, dem, G, \epsilon$**)**

1. /* Initialization stage */
2. $\forall a \in A, l(a) = \delta/c(a)$
3. **while** ($KCOMM \neq NULL$) **do** /* STAGE*/
4. $stageCounter++,\ phaseCounter = 1$
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. Build shortest path tree for the source S_i
9. $\forall C_k,\ s_k = S_i,\ dem_{IterRes}(C_k) = dem_{C_k}$
10. **while** newDL-lastDL < 1 and
11. $dem_{IterRes}(C_k) > 0$ for some k **do** /*STEP*/
12. If $\forall a \in P_{C_k}, l(a) \geq 1/c(a), l, C_k \in S_i$ then
13. P_{C_k} is not a shortest path
14. **if** NO connectivity for C_k **then**
15. $KCOMM = KCOMM \setminus \{C_k\}$
16. **for** $q = 1$ to r **do**
17. $c = \min_{a \in P_{C_q}} c(a)$
18. $f_{C_q} = \min(dem_{IterRes}, c)$
19. $f(P^j_{C_q}) = f(P^j_{C_q}) + f_{C_q}$ /*Update Curr Path*/
20. $dem_{IterRes} = dem_{IterRes} - f_{C_q}$
21. **end for**
22. $\forall a \in P_{C_i}, l(a) = l(a)(1 + \epsilon * \sum_{C_q : a \in P_{C_q}} \frac{f_{C_q}}{c(a)})$
23. $newDL = D(l)$
24. **end while** /* end of step */
25. **end for** /* end of iteration */
26. $phaseCounter++;$
27. **end while** /* end of phase */
28. $lastDL = newDL$
29. **end while** /* end of stage */
30. $\forall k = 1 \ldots K, \forall P \in P_k, f(P) = \frac{f(P)}{\log_{1+\epsilon} \frac{1+K\epsilon}{\delta}}$
31. $\forall k = 1 \ldots K, f(P_k) = \sum f(P^j_k)$
32. **Returns** per commodity k:
33. set of paths P_k and flows $\forall_{p^j_i \in P_i} f(p^j_i)$

Fig. 2. Approximation algorithm for the Max-Min fairness

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 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 22). 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, as thus need 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 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 30). 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 once instead of one shortest path calculation at a time. We will use this improvement [16] in the distributed implementation. The connectivity test per each commodity is done at this point, as well, 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. In our WMCMApprox 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 each stage termination condition ($D(l) > 1$ in the original primal problem) 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. But 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 23). At each stage at least one commodity is saturated and removed from the list $KCOMM$ since, at least, one link values is increased by a factor of $(1 + \epsilon)/c(a)$. This ensures the algorithm convergence.

Algorithm correctness and complexity Past analysis [2], [3], [16] showed the correctness of the maximum concurrent flow approximation algorithm and proved the dual-primal solution ratio, β/z, to be less than $1 + \xi$ for any $\xi > 0$. In addition, the following theorem was proved.

Theorem 1 *(Lemma* 3.2 *and Theorem* 3.1 *in [16]) There is an algorithm that computes a* $(1-\epsilon)^{-3}$*-approximation to the maximum concurrent flow in time* $O(\epsilon^{-2} m^2 \log\ m)$ *where* m *is the number of edges.*

Based on above theorem we provide the following analysis:

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

Proof: The analysis and proof in [2], [3], [16] hold for one stage of the WMCMApprox 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:

$$D(l_i) \quad \leq D(l_{i-1}) + \epsilon\alpha(l_i) \tag{5}$$

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. 5, the following holds

$$D(l_i) \leq \frac{D(l_{i-1})}{1 - \epsilon/\beta} \tag{6}$$

Since it holds for any stage that $D(l_{t_s}) \geq 1$, 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

$$K \leq D(l_t) \leq \frac{m\delta}{1-\epsilon} e^{\frac{\epsilon(t-1)}{\beta(1-\epsilon)}} \tag{7}$$

and a simple Algebraic manipulation yields

$$\beta \leq \frac{\epsilon(t-1)}{(1-\epsilon)\ln\ \frac{K \cdot (1-\epsilon)}{m\delta}} \tag{8}$$

Using the claim from [16] for each of the stages, summing up, and substituting in Eq. 8 we get

$$\beta/z < \frac{\epsilon}{(1-\epsilon)\ \ln\ (1+\epsilon) \cdot K} \cdot \frac{\ln\ \frac{1+K\epsilon}{\delta}}{\ln\ \frac{K(1-\epsilon)}{m\delta}} \tag{9}$$

By setting δ to be

$$\delta = \frac{1}{(1+K\epsilon)^{\frac{(1-\epsilon)}{(K-(1-\epsilon)}}} \cdot \left(\frac{K(1-\epsilon)}{m}\right)^{1+\frac{1-\epsilon}{K-(1-\epsilon)}} \tag{10}$$

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

Now it is left to show that the resulted rate vector is indeed max-min fair. This can be done by noticing the analogy between the operation of WMCM and WMCMApprox, and the proof of the correctness of WMCM in [1]. The proof can be found in the full version of this paper [17]. ■

B. Algorithm Implementation

We implemented the WMCMApprox algorithm using MATLAB. To illustrate the way the algorithm iterates, we provide the simple example of Figure 3. The capacity of each link is 1. There are four commodities, each with 1 unit of demand. All the links and paths are uni-directional. Commodities 2 and 4 has one path and its path ID is 1. Commodities 1 and 3 have 2 paths with IDs 1 and 2.

Table I 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 $0.3258 + 0.3438 + 0.3258 = 0.9954$. The calculated z for this stage is 0.3258 and the stage terminate 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 $KCOMM$. In the following stage, the algorithm iterates for commodity 1 between its two shortest paths till the saturation of both. The final maxmin vector rate for commodity 1 (path 1 and 2) commodities 2, 3 (path 1 and 2) and 4 is $\{1.6469\ 0.3258\ 0.3438\ 0.3258\}$. The final maxmin vector rate, when running the algorithm with $\epsilon = 0.1$ is $\{1.6585\ 0.3317\ 0.3317\ 0.3317\}$ for commodity 1 (path 1 and 2) commodities 2, 3 (path 1 and 2) and 4. Note that when the ϵ decreases the values are approaching the optimal weighted max-min vector (5/3,1/3,1/3,1/3).

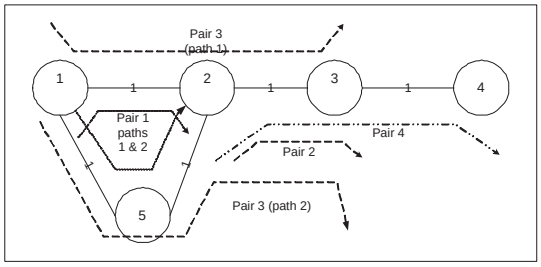

Fig. 3. Algorithm Iteration Example

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
length = {0.552 1.145 0.001 0.266 0.266}, z = 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	∞
length = {1.145 1.145 0.001 0.552 0.552}, z = 0.326, lastDL = 2.231					

TABLE I
SUMMARY OF THE EXECUTION

C. Weighted Max-min Fair - Distributed Approximation Algorithm

The distributed implementation of our algorithm is shown in Figure 4 for a source node. The code for the intermediate node or the destination node is omitted because of lack of space. We assume that each source router is familiar with the network topology, links capacities, the commodities for which it is serving as a source, and the ids of all other sources. In addition, the sources need to synchronize at the end of each

SOURCE-CODE($my\_id, MySrcCOMM, dem, G, \epsilon$**)**

1. $\forall a \in A, l(a) = \delta / c(a)$
2. **while** $(MySrcCOMM \neq NULL)$ **do** /* STAGE */
3. $stageCnt++,\ phaseCnt = 1$
4. $lastDL = 0; newDL = D(l)$
5. **while** $(newDL - lastDL < 1)$ **do** /* PHASE */
6. $\forall C_k, s_k = S_i\ dem_{ItrRes}(C_k) = dem(C_k)$
7. **while** $newDL - lastDL < 1$ and
8. $dem_{ItrRes}(C_k) > 0$ **do**
9. BldShrtPthTree
10. $(P_{C_k}, \forall a \in P_{C_k}, l(a) \leq 1/c(a), l, C_k \in S_i)$
11. $\forall C_k$ w/no connectivity
12. $MySrcCOMM = MySrcCOMM \setminus \{C_k\}$
13. **Parallel for** $q = 1\ to\ r$ **do**
14. $SndSRCAlc(nxtInP_{C_k}, my\_id, dem_{ItrRes}(C_k), P^j_{C_k})$
15. $WaitDestMsg(P^j_{C_k}, f(P^j_{C_q}), l)$
16. $dem_{ItrRes} = dem_{ItrRes} - f_{C_q}$
17. $newDL(l) = \sum_{a \in A} c(a) l(a)$
18. **end PAR for**
19. **end while** /* end of step */
20. $SndAl2AlSRCMsg(my\_id, l, phaseCnt, stageCnt)$
21. $GtAlSRCSYNMsg(l, phaseCnt, stageCnt)$
22. $newDL(l) = \sum_{a \in A} c(a) l(a)$
23. $phaseCnt++;$
24. **end while** /* end of phase */
25. $lastDL = newDL;$
26. **end while** /* end of stage */
27. $\forall k = 1 \ldots K, \forall P \in P_k, f(P) = \frac{f(P)}{\log_{1+\epsilon} \frac{1+K\epsilon}{\delta}}$
28. $\forall k = 1 \ldots K, f(P_k) = \sum f(P^j_k)$
29. **Returns** per commodity k: set of paths P_k and flows $\forall_{p^j_i \in P_i} f(p^j_i)$

Fig. 4. Distribute Max-min fair routing Algorithm (source code)

phase as explained later. By having all source nodes registered to a multicast group, one can simplify the synchronization process.

During a phase each source independently perform its procedure, iterating over the steps. In each step, the source sends an allocation request message over the pre-calculated shortest path tree. This message passes all intermediate nodes towards the destinations and collects the information about the bottleneck link over each shortest path. The destination node, upon source message arrival, sends Destination Acknowledge message with the appropriate bottleneck link information. Each intermediate router gets the destination message, updates its length and flow variables, and forwards it towards the source. Each source node, at the end of a phase, synchronizes with the other sources by exchanging the length information. This distributed algorithm can proceed until the network is saturated and each commodity gets 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 synchronization state before. This requires adding another message type that will be sent from the source node to the destination over 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 reset 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. The incremental algorithm for the case of adding a commodity is left for future research.

V. Concluding remarks

We presented a centralized and distributed approximated algorithm, which routes and allocates demands such that the max-min fairness criterion is achieved. However, it should be noticed that the provided solution is a local maximum for the max-min fair allocation rate vector. In a sequel work we provide the global solution for this problem.

References

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

[2] 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.

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

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

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

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

[7] 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, 2002.

[8] 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.

[9] J. M. Kleinberg, Y. Rabani, and E. Tardos, "Fairness in routing and load balancing," in *IEEE Symposium on Foundations of Computer Science*, 1999.

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

[11] 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.

[12] 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.

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

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

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

[16] G. Karakostas, "Faster approximation schemes for fractional multicommodity flow problems," in *ACM SODA*, 2002.

[17] M. Allalouf and Y. Shavitt, "Fast approximation algorithm for weighted max-min fairness with maximum flow routing," Dept. of Electrical Engineering – Systems, Tel Aviv University, Tech. Rep., 2004, technical Report EES2004-1.