On Max-min Fair Allocation for Multi-source Transmission

Geng Li
Yale University, USA
geng.li@yale.edu

Yichen Qian
Tongji University, China
92yichenqian@tongji.edu.cn

Y.Richard.Yang
Yale University, USA
yry@cs.yale.edu

ABSTRACT

Max-min fair is widely used in network traffic engineering to allocate available resources among different traffic transfers. Recently, as data replication technique developed, increasing systems enforce multi-source transmission to maximize network utilization. However, existing TE approaches fail to deal with multi-source transfers because the optimization becomes a joint problem of bandwidth allocation as well as flow assignment among different sources. In this paper, we present a novel allocation approach for multi-source transfers to achieve global max-min fairness. The joint bandwidth allocation and flow assignment optimization problem poses a major challenge with respect to nonlinearity and multiple objectives. We cope with this by deriving a novel transformation with simple equivalent canonical linear programming to achieve global optimality efficiently. Simulation results demonstrate that our approach is more max-min fair than other single-source and multi-source allocation approaches, meanwhile it outperforms others with substantial gains in terms of network throughput and transfer completion time.

1 INTRODUCTION

Max-min fair is a simple, classical and well-recognized sharing principle to define fairness in the field of data networks [10, 11]. In particular, it deals with the flow control problem in network traffic engineering (TE), where available resources (such as link bandwidth) are allocated among different traffic transfers [6–9]. Aiming at allocating rates to available links as evenly as possible, the max-min fair allocation is such that the result for the transfer with smallest sharing has been maximized over all feasible resource allocation solutions. Most of the existing works rely on the notion of bottleneck links, and a max-min fair allocation is conditioned by that each transfer must use at least one bottleneck link [13].

Data replication, a technique that amends both data availability and access efficiency, is emerging increasingly in recent datacenter networks and distributed filesystems [5, 12, 15]. To maximize network utilization and improve transmission performance, it is increasingly allowed to convey data in parallel from multiple sources for a transfer with multiple data replicas, *a.k.a* a multi-source transfer [1, 14]. However, a multi-source transfer will result in multiple flows, so multiple potential bottlenecks might be considered simultaneously to achieve transfer-level fairness. Existing TE approaches are no longer applicable because the allocation becomes a joint problem to optimally allocate each flow's bandwidth as well as to assign flows among the sources [6, 7, 9].

In this paper, we present a novel max-min fair allocation approach that jointly optimizes bandwidth allocation and flow assignment. The major challenge stems from that the direct formulation is nonlinear and multi-objective. We cope with this by deriving a novel transformation with simple equivalent canonical linear programming (LP) to achieve global optimality efficiently. We perform extensive simulations, which show that our approach leads to better performance on network throughput with a gain of up to 52% and reduces transfer completion time by up to 44%, compared to other single-source and multi-source allocation approaches.

2 NETWORK MODEL AND PROBLEM FORMULATION

Consider a telecommunications network composed of a set of nodes and a set of links $\mathcal{L}$. The capacity of link L_j ($j \in [1, M]$) is defined as C_j. Suppose there are a number of data transfers, each of which may come from multiple sources. Thus each transfer i ($i \in [1, N]$) is assigned with a set of flow paths $\{P_{i1}, ..., P_{ik}\}$, and each such path is identified with the set of links that it traverses, *i.e.,* $P_{ik} \subseteq \mathcal{L}$. Now let r_i denote the data rate of transfer i, which is the sum bandwidth of the constituent flows from all its sources. We use a variable set $\mathcal{X}_i = \{x_{i1}, ..., x_{ik}\}$ to express the flow assignment proportions from different sources for transfer i, and $\sum_{k=1}^{K_i} x_{ik} = 1$, where K_i is the total source number of transfer i.

We are interested in solving the joint bandwidth allocation and flow assignment problem, *i.e.,* finding the transmission rate r_i of each transfer, together with the assignment proportions $\mathcal{X}_i$ from all its sources. The solution is required to provide a fair and efficient allocation result, and its precise objective and constraints are described as below.

Objective. When computing allocated bandwidth, our goal is to maximize network utilization while in a max-min fair manner. A vector of transfer rate allocations $\{r_i\}$ is said to be max-min fair if, for any other feasible allocation $\{r'_i\}$, the following has to be true: $\forall r'_p > r_p$ for the data transfer p, there exists another transfer q such that $p, q \in [1, N]$, $r'_q < r_q$, $r_q \le r_p$. In other words, increasing some components must be at the expense of decreasing some other existing smaller or equal components.

Constraints. The constraints of this problem are given as below. Constraint (1) is called the capacity constraint, which assures that for any link L_j, its load does not exceed its capacity C_j. Constraint (2) promises the sum fractions of a transfer from all available sources equal to 1.

$$s.t. \quad \sum_{L_j \subseteq P_{ik}} r_i \cdot x_{ik} \le C_j \qquad \forall j, \tag{1}$$

$$\sum_{k=1}^{K_i} x_{ik} = 1 \qquad \forall i, \tag{2}$$

$$0 \le x_{ik} \le 1 \qquad \forall i, k. \tag{3}$$

Figure 1 shows an example of the network consisting of six links $\{L_1, ..., L_6\}$, with capacities $\{8, 5, 4, 5, 7, 6\}$ respectively. Assume all bandwidth numbers are in *Gbps* here. Suppose there are three data transfers, among which transfer 1 and 2 have a single data source while transfer 3 has two available sources. In total, there are four potential flows in the network, including $f_1 : A \to C \to B \to D$,

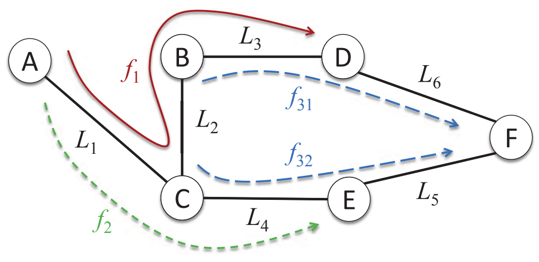

Figure 1: An example with 3 transfer requests. Transfer 3 comes from two feasible sources B and C, corresponding to two parallel flows f_{31} and f_{32}.

$f_2 : A \rightarrow C \rightarrow E$, $f_{31} : B \rightarrow D \rightarrow F$ and $f_{32} : C \rightarrow E \rightarrow F$. Given the topology and values of link capacities, we aim at finding the max-min fair solution of the rate vector $\{r_1, r_2, r_3\}$ and the flow assignment $\{x_{31}, x_{32}\}$ for transfer 3.

3 APPROACH SPECIFICATION

One of the biggest challenges for multi-source transmission stems from changing the optimized object from an individual 5-tuple flow into a group of flows that belong to the same transfer. Hence, multiple potential bottlenecks might be considered simultaneously. Accordingly, we design a MultiSource Water-Filling (MS-WF) algorithm that jointly computes bandwidth allocation and flow assignment while providing global max-min fairness. The key to MS-WF algorithm is a novel transformation with simple equivalent canonical LP. In this section, we first briefly describe the traditional water-filling algorithm and its limitation. Then we present the MS-WF with one multi-source transfer for clear illustration. At last we provide the generic solution that optimizes arbitrary flow transfers.

3.1 Flow-link Mapping Matrix and Traditional Water-Filling Algorithm

The proposed MS-WF algorithm is fundamentally based on a flow-link mapping matrix (FL matrix) with solvable variables. Here we define the FL matrix, and illustrate the traditional water-filling algorithm with this matrix.

Definition 1 (Flow-link Mapping Matrix)**.** The flow-link mapping matrix *(FL matrix)* $\{fl_{ij}\}$ expresses the flow paths and the traffic shares of each transfer in matrix form. The matrix element fl_{ij} is defined as the proportion of flow i in its belonging transfer that traverses link L_j.

Water-Filling (WF) Algorithm. The traditional WF algorithm can compute the bandwidth allocations for single-source transmission. Using the FL matrix as an input, we first denote the saturated average bandwidth allocation as $\tau_j = C_j/n_j$, where n_j is the total number of flows that use link L_j. The WF algorithm iteratively finds the minimum τ^* and the corresponding bottleneck link L_{j^*}. Set the bandwidth of the flows that use link L_{j^*} to τ^*. Then update the FL matrix by subtracting those flows and the bottleneck link to calculate a new set of $\{\tau_j\}$. Such process iterates until all transfers obtain their allocated rates, *i.e.,* all flows have bottleneck links.

	L_1	L_2	L_3	L_4	L_5	L_6
f_1	1	1	1	0	0	0
f_2	1	0	0	1	0	0
f_{31}	0	0	1	0	0	1
C_j	8	5	4	5	7	6
n_j	2	1	2	1	0	1
τ_j	4	5	2	5	*NA*	6

(a)

	L_1	L_2	L_4	L_5	L_6
f_2	1	0	1	0	0
C_j	6	3	5	7	4
n_j	1	0	1	0	0
τ_j	6	*NA*	5	*NA*	*NA*

(b)

Figure 2: An example of the FL matrix for single-source transmission, where transfer 3 only uses one source. C_j is the bandwidth capacity, $n_j = \sum_i fl_{ij}$ is the total number of flows that use link L_j, and $\tau_j = C_j/n_j$ is the saturated average bandwidth share. (a) Illustration of the first iteration, where L_3 is found as the bottleneck link with minimum τ_j. (b) Illustration of the second iteration with the updated FL matrix, where L_4 is then found as the bottleneck link.

Example. Consider a single-source version of Figure 1, where we assume transfer 3 only uses source B, so there are 3 flows over the network. Figure 2 illustrates the allocation algorithm for this single-source example. In the first iteration, L_3 is first saturated because τ_3 is of the minimum value $\tau^* = 2$. We allocate the bandwidth of f_1 and f_{31} that traverse L_3 to 2, and then remove the rows of f_1, f_{31} and column L_3. The FL matrix is updated accordingly for the next iteration, where link L_4 is then found as the bottleneck and bandwidth of f_2 is set to 5. By this point, the ultimate solution to the rate allocation for the 3 transfers is (2, 5, 2).

The traditional WF algorithm succeeds in obtaining the max-min fair allocation for single-source transmission, yet it is incapable of dealing with multi-source transfers. This principally stems from the fact that the algorithm is based on flows, rather than on transfers. For the example in Figure 1, transfer 3 will have double weights by using water-filling algorithm, which is unfair to the others. A naïvely improved solution is to normalize the weight of each transfer, and to assign an equal share to the flows from different sources. With regard to the example, the elements for f_{31} and f_{32} become 1/2 and 1/2 in the FL matrix, such that each transfer has the same sum weight of 1. The allocation result turns into (8/3, 10/3, 3), which is slightly better than the single-source cases, which are (2, 5, 2) for using only source B and (2.5, 4, 2.5) for using only source C. However, as we will soon learn, equally sharing the flow weight is still not the optimal solution. *The transfer-level max-min fair allocation is conditioned by the optimal flow assignment.*

3.2 MS-WF Algorithm with One Multi-source Transfer

Given the FL matrix and its application, now let's consider a more complex case by adding just one multi-source transfer into the network. We assume there are K_m available sources for the particular transfer m, though the flow assignment $\mathcal{X}_m$ is unknown and needs to be solved. The MS-WF algorithm continues to use the FL matrix as an input, but the elements are no longer 0 and 1 as in the single-source case. Instead, the matrix elements related to transfer m are replaced by the unknown variables in $\mathcal{X}_m$. The MS-WF algorithm

Algorithm 1 MS-WF Algorithm with one multi-source transfer

```
Input:
    Flow-link mapping matrix: $FL = \{fl_{ij}\}$;
    Capacity of each link: $\{C_j\}, 1 \leqslant j \leqslant M$;
Output:
    Transmission rate of each transfer: $\{r_i\}, 1 \leqslant i \leqslant N$;
    Flow assignment of transfer $m$: $\mathcal{X}_m = \{x_{m1}, \ldots, x_{mk}\}$;
• Step 1:                                              ▷ Initiation
 1: $r_i \leftarrow 0, \quad \forall i = 1, \ldots, N$;
 2: $\mathcal{L} \leftarrow \{L_1, L_2, \ldots, L_M\}$;
• Step 2:                 ▷ Calculate the saturated average bandwidth
 3: $n_j(\mathcal{X}_m) \leftarrow \sum_i fl_{ij}, \quad \forall j \in \mathcal{L}$;
 4: $\tau_j(\mathcal{X}_m) \leftarrow C_j / n_j(\mathcal{X}_m), \quad \forall j \in \mathcal{L}$;
• Step 3:                          ▷ Find the bottleneck fair share $\tau^*$
 5: if $min\{\tau_j(\mathcal{X}_m)\}$ is a constant then
 6:     $\mathcal{X}^* \leftarrow \varnothing$;
 7: else
 8:     $\mathcal{X}^* \leftarrow \mathcal{X}_m | max\ min\{\tau_j(\mathcal{X}_m)\}$;
 9: end if
10: $\tau^* \leftarrow min\{\tau_j(\mathcal{X}_m)\}$;
• Step 4:                    ▷ Set data rate and update the FL matrix
11: $\mathcal{L}_{j^*} \leftarrow \{L_j | \tau_j(\mathcal{X}_m) = \tau^*\}$;
12: $\mathcal{L} \leftarrow \complement_{\mathcal{L}_{j^*}} \mathcal{L}$;
13: for $i | P_i \cap \mathcal{L}_{j^*} \neq \varnothing$ do
14:     $r_i \leftarrow r_i + \tau^*$;
15:     Remove $f_i$ from $FL$;
16: end for
17: Update $FL$;
• Step 5:                                              ▷ Iteration
18: if No transfers left then
19:     return $\{r_i\}$ and $\mathcal{X}_m$;
20: else
21:     goto Step 2.
22: end if
```

with one multi-source transfer (**Algorithm 1**) can be described in the following high-level steps:

(1) Start from zero allocation with the whole link set, and build the FL matrix with variables $\mathcal{X}_m$.
(2) Calculate the saturated average bandwidth $\tau_j(\mathcal{X}_m)$ on each link L_j.
(3) Find the bottleneck fair share $\tau^* = min\{\tau_j(\mathcal{X}_m)\}$ by solving $\mathcal{X}^* = \mathcal{X}_m | max\, min\{\tau_j(\mathcal{X}_m)\}$.
(4) Set the data rate to τ^* for the flows that traverse the bottleneck links, and update the FL matrix by removing those flows and links.
(5) If there are no transfers left then stop, otherwise return to Step 2.

In Step 2, similar to the traditional WF algorithm, we first calculate $n_j(\mathcal{X}_m)$ by summarizing the elements of column j in the FL matrix. Here $n_j(\mathcal{X}_m)$ denotes the number of transfers that use link L_j. Due to the fact that transfer m comes from multiple parallel flows, there might be only a fraction of the transfer using link L_j. As a result, n_j becomes a function of $\mathcal{X}_m$ instead of an integer value as in the single-source case. Next, we compute the average bandwidth $\tau_j(\mathcal{X}_m) = C_j / n_j(\mathcal{X}_m)$, which is also a function of $\mathcal{X}_m$.

In Step 3, given $\{\tau_j(\mathcal{X}_m)\}$ on the current link set, one or several bottleneck links are found. Specifically, since all the variables

	L_1	L_2	L_3	L_4	L_5	L_6
f_1	1	1	1	0	0	0
f_2	1	0	0	1	0	0
f_{31}	0	0	x	0	0	x
f_{32}	0	0	0	$1-x$	$1-x$	0

C_j	8	5	4	5	7	6
n_j	2	1	$1+x$	$2-x$	$1-x$	x

τ_j	4	5	$\frac{4}{1+x}$	$\frac{5}{2-x}$	$\frac{7}{1-x}$	$\frac{6}{x}$

Figure 3: An example of MS-WF, where transfer 3 comes as two flows. x is the flow assignment variable to be calculated. τ_j in orange cell is found as the minimum bottleneck share.

$\mathcal{X}_m$ are within a certain range $[0,1]$, $min\{\tau_j(\mathcal{X}_m)\}$ is sometimes a constant value. In that case, no flow assignment variable is calculated, *i.e.,* $\mathcal{X}^* = \varnothing$. Otherwise, the flow assignment is determined by $\mathcal{X}^* = \mathcal{X}_m | max\, min\{\tau_j(\mathcal{X}_m)\}$. We use τ^* to denote the minimum bandwidth share, and the set of $\{L_{j^*}\}$ is found as the bottleneck links.

Back to the example in Figure 1, transfer 3 has two available sources to access, resulting in two parallel flows f_{31} and f_{32}, respectively. For simplicity, we use only one variable x to denote the proportion of f_{31}, and that of f_{32} will thus be $1-x$. Figure 3 illustrates the FL matrix, followed by the saturated average bandwidths $\{\tau_j(\mathcal{X}_m)\}$.

From the example, we can tell that *Step 3, finding $\mathcal{X}^*$ that maximizes $min\{\tau_j(\mathcal{X}_m)\}$, is the major challenge in MS-WF.* Accordingly, it is to find $x^* = x | max\, min(4, 5, 4/(1+x), 5/(2-x), 7/(1-x), 6/x)$ in Figure 3. First, as formulated in **Problem 1**, it is a nonlinear programming problem, which can not be solved directly. Second, even if we can find a linear expression, we still need long sequences of LPs for the max-min objective (multi-objective), which is computationally intense in real implementation.

Problem 1 (The optimization problem in Step 3)**.**

$$max \quad min\{\tau_j(\mathcal{X}_m)\}, \tag{4}$$

$$s.t. \quad \sum_{k=1}^{K_m} x_{ik} = 1 \qquad x_{ik} \in \mathcal{X}_m, \tag{5}$$

$$0 \leq x_{ik} \leq 1 \qquad \forall i, k. \tag{6}$$

To cope with this, we transform this nonlinear optimization problem into a canonical form of LP problem based on **Theorem 1**. As a result, the equivalent canonical LP problem is expressed as **Problem 2**, which can be solved efficiently with limited computational complexity.

Problem 2 (The equivalent LP problem in Step 3)**.**

$$min \quad t \tag{7}$$

$$s.t.\quad t \geq \tau'_j(\mathcal{X}_m) \qquad \forall j, \tag{8}$$

$$\sum_{k=1}^{K_m} x_{ik} = 1 \qquad x_{ik} \in \mathcal{X}_m, \tag{9}$$

$$0 \leq x_{ik} \leq 1 \qquad \forall i, k. \tag{10}$$

Theorem 1. *Problem 1 is equivalent to Problem 2 as a canonical LP problem, where $\tau'_j(\mathcal{X}_m) = 1/\tau_j(\mathcal{X}_m)$.*

Proof. Given an arbitrary instance of the FL matrix, $\tau_j(\mathcal{X}_m)$ satisfies two conditions: i) $\tau_j(\mathcal{X}_m) \geq 0$, and ii) the inverse of $\tau_j(\mathcal{X}_m)$ is a linear function of $\mathcal{X}_m$. So we let $\tau'_j(\mathcal{X}_m) = 1/\tau_j(\mathcal{X}_m)$, and the objective of $max\,min\{\tau_j(\mathcal{X}_m)\}$ is then equivalent to $min\,max\{\tau'_j(\mathcal{X}_m)\}$, which becomes linear accordingly. Next, we introduce a temporary variable $t = max\{\tau'_j(\mathcal{X}_m)\}$, and use a sequence of inequality constraints $t \geq \tau'_j(\mathcal{X}_m)$ for all j to express t. Since $\tau'_j(\mathcal{X}_m)$ is a linear function of $\mathcal{X}_m$, the constraint (8) as a set of inequalities is also linear. In the end, the optimization problem in Step 3 (**Problem 1**) turns into an equivalent canonical LP problem (**Problem 2**), where the decision variables to be solved are the flow assignment set $\mathcal{X}_m$ and t. □

As a result, the optimization problem of the example in Figure 3 is well transformed into a simple equivalent LP problem, which is expressed as below.

$$min \quad t \tag{11}$$

$$s.t. \quad t \geq \frac{1}{4}, \tag{12}$$

$$t \geq \frac{1}{5}, \tag{13}$$

$$4t - x \geq 1, \tag{14}$$

$$5t + x \geq 2, \tag{15}$$

$$7t + x \geq 1, \tag{16}$$

$$6t - x \geq 0, \tag{17}$$

$$0 \leq x \leq 1. \tag{18}$$

The results of the above LP come out as $x = 1/3$ and $t = 1/3$. Then $\tau^* = 1/t = 3$ is the minimum fair share, and L_3 and L_4 are the bottleneck links that are saturated in this iteration. We set $r_1 = r_2 = 3$ as the bandwidth of f_1 and f_2, and $r_3 = 3$ as the sum bandwidth of f_{31} and f_{32}. Meanwhile, the data volume assignment of transfer 3 concludes with $1/3$ from source B and $2/3$ from source C. The final rate allocation to the 3 transfers are (3, 3, 3), which is more max-min fair than the allocation (2, 5, 2) where transfer 3 uses only source B (as in Figure 2), as well as the allocation (2.5, 4, 2.5) where transfer 3 uses only source C. In addition, if we don't consider the impact of date volume and assume each transfer has the equal volume of $3Gbits$, then MS-WF outperforms the single source approaches in terms of the average completion time (MS-WF: (3/3+3/3+3/3)/3=1, source B: (3/2+3/5+3/2)/3=1.2) and source C: (3/2.5+3/4+3/2.5)/3=1.05), as well as total completion time (MS-WF: 3/3=1, source B: 3/2=1.5 and source C: 3/2.5=1.2).

3.3 Generic MS-WF

Having solved the preliminary instance with one multi-source transfer, now we consider the generic MS-WF with arbitrary transfer combinations. Here the variables become the flow assignments $\{\mathcal{X}_i\}$ ($i \in [1, N]$) for all transfers. Except for the transfers with single source, whose $\mathcal{X}_i = \{1\}$ for all the time, the rest of $\{\mathcal{X}_i\}$ are to be computed by MS-WF. The main challenge is that the flow assignments of different transfers correlate to each other and can not be calculated independently. One transfer's assignment plan affects another's optimal decision. *Therefore, the max-min fair allocation requires joint calculation for all flow assignments.*

The generic MS-WF mainly follows the procedures in **Algorithm 1**. Exceptionally, we put all sets of the variables $\{\mathcal{X}_i\}$ into the FL matrix, such that τ_j turns into a function of $\{\mathcal{X}_1, ..., \mathcal{X}_N\}$. By the same token, we transform the nonlinear optimization problem in Step 3 into an equivalent canonical LP problem with the help of one additional decision variable t. In each iteration, parts of the flow assignment sets are solved by LP based on **Theorem 2**. Then we plug the values into the FL matrix and remove them from $\{\mathcal{X}\}$. Continue iterating until all flow assignment variables $\mathcal{X}_i$ are determined. Finally, we sum up the constituent flow rates as the multi-source transfer rate, *i.e.,* $r_i = \sum_{k=1}^{K_i} r_{ik}$, where K_i is the total source number of transfer i.

Theorem 2. *Multiple sets of variables $\mathcal{X}_i$ can be jointly calculated by MS-WF.*

We simplify this theorem by a small scale instance that involves only two multi-source transfers, and each of them has only one variable. Specifically, we denote x_1 as the assignment for one transfer, and x_2 for the other. Formally, we prove the following lemma.

Lemma 1. *The optimal x_1^* and x_2^* can be jointly calculated by the LP in MS-WF.*

Proof. Each iteration is divided into 5 (types of) cases.

Case 1: One bottleneck is found as $t = \tau'_{j1}$. Since it's a constant, neither x_1^* or x_2^* is determined in this iteration.

Case 2: One bottleneck is found as $t = \tau'_{j1}(x_1)$. Then we use the constraint $0 \leq x_1 \leq 1$ to find an intersection point x_1^* that minimizes t. Substitute x_1^* and solve x_2^* accordingly.

Case 3: Two bottlenecks are found as $t = \tau'_{j1}(x_1)$ and $t = \tau'_{j2}(x_1)$. Since they are not related to x_2, we can simply make $\tau'_{j1}(x_1) = \tau'_{j2}(x_1)$ to solve the intersection point x_1^*. Substitute x_1^* and solve x_2^* accordingly.

Case 4: Two bottlenecks are found as $t = \tau'_{j1}(x_1, x_2)$ and $t = \tau'_{j2}(x_1)$. We first decompose $\tau'_{j1}(x_1, x_2)$ as $\tau'_{j1}(x_1, x_2) = \tau'_{j1}(x_1) + \tau'_{j1}(x_2)$ with respect to linearity. Solve x_2^* first from $\tau'_{j1}(x_2)$ via $0 \leq x_2 \leq 1$ as in Case 2. Substitute $\tau'_{j1}(x_2^*)$ and solve x_1^* as in Case 3.

Case 5: Two bottlenecks are found as $t = \tau'_{j1}(x_1, x_2)$ and $t = \tau'_{j2}(x_1, x_2)$. Make $\tau'_{j1}(x_1, x_2) = \tau'_{j2}(x_1, x_2)$ to solve the relation function $x_1^* = f(x_2^*)$. Ignore the current two bottlenecks and find the next one(s), until Case 2, 3, or 4 happens. Solve either x_1^* or x_2^* and substitute the result into $x_1^* = f(x_2^*)$ to solve the other.

To summarize, in any cases, multiple sets of variables are solvable by the canonical LP in MS-WF. □

The MS-WF algorithm is scalable and extensible for more complex use cases. For instance, differentiated qualities of service lead to transfers with variations in priority or other requirements, such that MS-WF is capable of supporting weighted fair allocation by taking priority factors into account. The max-min fair principle can also be applied to different optimization objectives (*e.g.,* transfer completion time), and the adaption of MS-WF with consideration of data size can likewise yield the optimal results for multi-source transfers. The computation complexity is significantly reduced in MS-WF by transforming a nonlinear multi-objective problem into a single LP, and it can be further reduced by removing the redundant constraints in implementation.

4 PERFORMANCE EVALUATION

To evaluate how MS-WF algorithm works on a large-scale network, we implement a flow-level datacenter network simulator.

4.1 Simulation Methodology

Topologies: We conduct our experiments by emulating a 3-tier datacenter network topology with 8:1 oversubscription. The topology contains 64 servers, whereby each edge link is of $1Gbps$ capacity, and aggregated link is of $10Gbps$ capacity.

Workloads: We synthesize a stream of transfer requests with a total number of 1000. The request arrival is modeled as a Poisson process, where the arrival rate λ is defined as the average number of new transfers per time slot. We set the slot length as one second for fast simulation. A transfer has multiple sources with probability ρ. A multi-source transfer is assumed to have a random number of replicas between [2,5], and the replicas are randomly placed in servers. In simulations, we do not consider the fluctuation of transfer size, and assume all transfers have an uniform size of V.

Performance metrics: We use *network throughput* and *average transfer completion time* to show the improvements of MS-WF over other approaches.

Alternative approaches: We compare the following bandwidth allocation approaches, each of which adopts the traditional WF algorithm.

- **Best-source:** This approach selects a best replica source based on the algorithm in [14] for multi-source transfers.
- **Equal-share:** This approach equally splits the transfer across different sources. For example, if a transfer has 3 replicas, each replica will send 1/3 of the data.
- **Random-source:** This approach randomly selects an available source to transmit data.

4.2 Simulation Results

Figure 4 shows the simulation results of network throughput for various approaches. Here we set the arrival rate $\lambda = 2$ and the data size $V = 10Gbits$ for all of the 1000 transfers. As the results show, Random-source approach disregards the source dissimilarity, therefore performs the worst with a constant throughput value. Equal-share approach takes advantage of source diversity simplistically. When the diversity is limited to a small number of multi-source transfers (at low multi-source probabilities), equal flow sharing approximates the optimal assignment, and thus obtains the similar performance as MS-WF. But as the multi-source proportion increases, there are more flows entering the network. The effect of "bad flows" enlarges, and hence drags down the overall throughput improvement. Accordingly, Best-source approach begins to outperform Equal-share. By jointly optimizing the bandwidth allocation

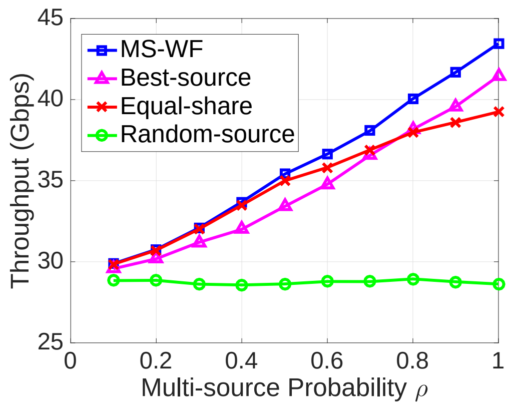

Figure 4: Network throughput vs. multi-source probability ρ, where the arrival rate $\lambda = 2$ and the data size $V = 10Gbits$.

and flow assignment, MS-WF achieves a much higher throughput than the others, resulting in higher utilization of the network. When all the transfers have multiple sources ($\rho = 1$), MS-WF obtains a substantial throughput gain of up to 52% compared to the single-source transmission.

Figure 5 compares the transfer completion time versus three factors respectively: the multi-source probability ρ, the transfer size V and the transfer arrival rate λ. It is shown that, MS-WF achieves the smallest transfer completion time across all parameter configurations. This improvement is mainly derived from a more max-min fair allocation by MS-WF.

Figure 5(a) focuses on the impact of multi-source probability ρ. Approximately, ρ equals to the proportion of multi-source transfers. The gap between Random-source and the other approaches verifies that, leveraging multiple sources is more efficient for data transmission. Moreover, the sustained decline in completion time with the multi-source probability implies that, the multi-source transmission obtains more performance gains by placing more replicas in the network. Compared to single-source transmission, MS-WF reduces the average completion time by up to 44%.

Figure 5(b) shows the relationship between completion time and the transfer size V. As the transfer size increases, the transfer backlog starts to cause more bottleneck links, which leads to the degradation of transmission rates. Therefore the completion time increases superlinearly along with the transfer size for all the allocation approaches. But the almost linear completion time growth of MS-WF suggests that, by completing transfers as quick as possible, MS-WF is capable of optimizing data transfers for small files as well as large files.

Figure 5(c) illustrates the impact of transfer arrival rate λ. As expected, at higher rates, the number of transfers over the network potentially increases, and links are more likely to become congested. Accordingly, the performance degrades quickly for all the approaches except MS-WF. The smaller growth in completion time demonstrates that, by effectively avoiding the congestion point, MS-WF manages to handle relatively a larger amount of traffic without degrading the performance.

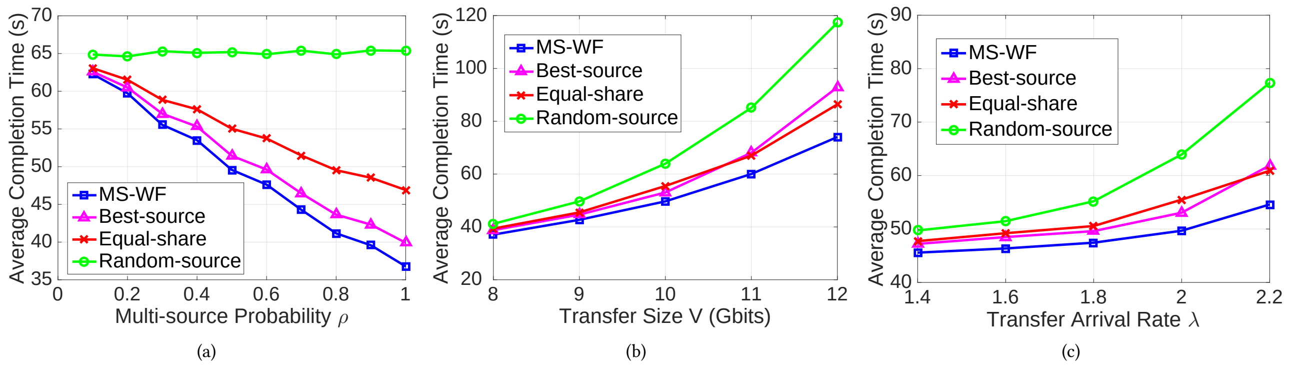

Figure 5: Impact of (a) the multi-source probability ρ, (b) the data size V and (c) the transfer arrival rate λ on average transfer completion time. In each subfigure, we adjust one factor and fixed the other two. The three default values are $\rho = 0.5$, $V = 10$ and $\lambda = 2$.

5 RELATED WORK

Distributed filesystems. Several high-performance distributed filesystems with sufficient data replicas have been developed, including GFS [5], HDFS [15] and Quantacast File System [12]. Leveraging software-defined networking (SDN), Mayflower [14] performs global optimizations to make intelligent replica selection and flow scheduling decisions based on both filesystem and network information. Nevertheless, current solutions focus heavily on best replica selection and data replication placement, instead of multi-source transmission as in our work. As shown in the proceeding sections, single-source transmission fails to achieve transfer-level max-min fairness, therefore provides sub-optimal performance.

Coflow scheduling. The works that schedule parallel flows have been developed to optimize transfers at the level of coflow rather than individual ones. Coflow [2], Varys [3] and Barrat [4] improve application-level performance by minimizing coflow completion times and guaranteeing predictable completions. However, their basic assumption is that the flows are streamed for different data and the volume of each flow is designated in advance, so they can easily predict the completion time and allocate the rate to meet their deadlines. The improvement of our approach over them is tat the flow volume assignment is jointly optimized with bandwidth allocation to achieve global optimality.

6 CONCLUSION

We present a novel max-min fair allocation approach for multi-source transmission which conveys data in parallel from multiple sources and dynamically adjusts the flow volumes to maximize network utilization. The allocation relies on a MultiSource Water-Filling algorithm that jointly computes the bandwidth allocation and flow assignment with simple equivalent canonical LP to achieve global optimality. Extensive simulations validate that, compared to other single-source and multi-source allocation approaches, our approach achieves a better throughput gain of up to 52% and decreases transfer completion time by up to 44% for large-scale transfers. We believe this approach is applicable to various traffic management systems that orchestrate arbitrary bulk transfers. Developing such systems will be the next step of this research.

REFERENCES

[1] M. Chowdhury, S. Kandula, and I. Stoica. Leveraging endpoint flexibility in data-intensive clusters. In *ACM SIGCOMM CCR*, volume 43, 2013.
[2] M. Chowdhury and I. Stoica. Coflow: A networking abstraction for cluster applications. In *Proceedings of HotNet*, 2012.
[3] M. Chowdhury, Y. Zhong, and I. Stoica. Efficient coflow scheduling with varys. In *ACM SIGCOMM CCR*, 2014.
[4] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron. Decentralized task-aware scheduling for data center networks. In *ACM SIGCOMM CCR*, 2014.
[5] S. Ghemawat, H. Gobioff, and S.-T. Leung. The google file system. In *ACM SIGOPS operating systems review*, 2003.
[6] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven wan. In *SIGCOMM CCR*, 2013.
[7] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al. B4: Experience with a globally-deployed software defined wan. *SIGCOMM CCR*, 2013.
[8] X. Jin, Y. Li, D. Wei, S. Li, J. Gao, L. Xu, G. Li, W. Xu, and J. Rexford. Optimizing bulk transfers with software-defined optical wan. In *Proceedings of SIGCOMM 2016 Conference*.
[9] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, et al. Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In *SIGCOMM CCR*, 2015.
[10] Q. Ma, P. Steenkiste, and H. Zhang. Routing high-bandwidth traffic in max-min fair share networks. In *Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, pages 206–217, 1996.
[11] D. Nace and M. Pioro. Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial. *IEEE Communications Surveys & Tutorials*, 10(4):5–17, 2009.
[12] M. Ovsiannikov, S. Rus, D. Reeves, P. Sutter, S. Rao, and J. Kelly. The quantcast file system. *Proceedings of the VLDB Endowment*, 6(11), 2013.
[13] B. Radunovic and J. Y. L. Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Transactions on Networking*, 15(5):1073–1083, 2007.
[14] S. Rizvi, X. Li, B. Wong, F. Kazhamiaka, and B. Cassell. Mayflower: Improving distributed filesystem performance through sdn/filesystem co-design. In *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, 2016.
[15] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, 2010.