Blossom: Content Distribution using Inter-Datacenter Networks

Yangyang Li, Haiyong Xie, Yong Liao
Innovation Center
China Academy of Electronics and Information Technology, Beijing, 100041, China
Email: liyangyang@live.com, haiyong.xie@ieee.org, liaoyong@live.com

***Abstract*—Cloud service providers are building out geographically distributed networks of datacenters around the world. It is customary for cloud service providers to distribute their data replicas at multiple geographic locations to mitigate user latency and to increase service availability. In this paper, we treat the content distributed from one datacenter to multiple datacenters as a multicast session. We investigate the problem of maximizing the capacity utilization of inter-datacenter networks while maintaining fairness among multiple multicast sessions. A bandwidth allocation algorithm based on max-min fairness is developed, named *Blossom*. *Blossom* leverages a fully polynomial time approximation scheme to accelerate the bandwidth allocation, while achieving an approximation that is $(1-\epsilon)$-optimal. Through trace-driven simulation, we show that our approach is substantially more efficient than prior work.**

I. Introduction

Nowadays, it becomes a common practice for cloud service providers to build geographically distributed datacenters worldwide [1], [2]. In order to mitigate user latency and to increase service availability, it is necessary to distribute the contents to multiple physical locations closer to users [3], [4]. With the proliferation of bandwidth-intensive applications, such as online video and file sharing, the scale of content distribution among these datacenters grows exponentially [5]. The management of inter-datacenter traffic has drawn much attention in academia.

Several recent research works [6], [7] introduced centralized traffic engineering for data transfers among datacenters. With supporting numerous types of applications in mind, they did traffic control at a flow level, which considers one to one flows between datacenter pairs, i.e., unicast flows. In this paper, we focus on one specific kind of application. We need to distribute the same content from one datacenter to multiple datacenters. Using multiple unicast flows to send such content is inefficient, because some network links carry multiple copies of the same content. A more natural way is to treat a content-distributing session as a multicast session, which consists of a source datacenter and a set of destination datacenters.

The question raised in papers [6] and [7] becomes how to improve the capacity utilization of inter-datacenter networks by using multicast sessions. A straightforward solution to improve the capacity utilization is to distribute the content along multiple multicast trees [8], [9]. However, the sharing nature of inter-datacenter networks makes this problem more challenging. Large volumes of contents need to be delivered from different source datacenters to the distinct sets of destination datacenters simultaneously.

There is a trade-off between achieving optimal capacity utilization and maintaining fairness among multiple multicast sessions. On one hand, towards maximizing capacity utilization, the available bandwidth should be utilized as much as possible. As the multicast sessions with more members could potentially use more spanning trees, they would dominate the bandwidth allocation. That is to say, they would probably obtain higher session rates. On the other hand, if we aim to maintain fairness, we need to allocate the same proportion of demand rate to each session. Capacity utilization would be low, since the proportion is constrained by the session which is saturated first, despite the fact that there still be available bandwidth that could be allocated to other sessions.

Max-min fair allocation fits this bill. By max-min fairness, we can increase the rate of any sessions without decreasing the allocation of a session with smaller or equal rates. This strategy is well studied in packet networks [10], [11], data center networks [12], and even in inter-datacenter networks [6], [7]. However, the authors in [6], [7] only consider unicast flows across datacenters, multicast makes it an intractable problem to solve. Traditionally, the max-min fair allocation problem is formulated as a path-flow linear program (LP), and standard LP solvers (e.g., CLP 1.13.3 [13]) are most commonly used [11]. However, the number of paths, i.e., spanning trees in our case for a multicast session, can be as large as 16807 [14] in a fully connected 7 datacenter inter-datacenter network. The exponential number of variables in the LP problem makes it is time-consuming and storage-expensive to use the standard LP solvers.

In this paper, we view that massive content distribution among datacenters as a bandwidth allocation problem for multiple multicast sessions. Rather than using the standard LP solvers, we are interested in developing an approximation algorithm that can solve the problem in polynomial time while achieving $(1-\epsilon)$-optimal. In particular, we develop an algorithm called *Blossom*, which allocates bandwidth to multiple multicast sessions in a weighted max-min fair way. Our trace-driven simulation shows that *Blossom* outperforms existing multi-concurrent multicast solutions [8].

The remainder of this paper is organized as follows. We formulate the max-min fair multi-tree multicast problem in Sec. II. In Sec. III, we develop a fully polynomial time ap-

978-1-5090-1328-9/16/$31.00 ©2016 IEEE

proximation algorithm and prove the correctness of weighted max-min fairness. Amazon EC2 trace-based simulation results are given in Sec. IV, followed by related work in Sec. V and our conclusion in Sec. VI.

II. Model and Definitions

In this section, we describe our system model and clarify the definition of max-min fairness in our context.

A. System Model

Let us consider a cloud service provider which runs multiple datacenters located across geographically distributed regions. These datacenters form a complete directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ indicates the set of datacenters, and $\mathcal{E}$ indicates the set of directed links inter-connecting them. For each directed link $e \in \mathcal{E}$, we use c_e to denote the capacity of directed link e, which is the maximum available rate of packet transmission on e.

Let $\mathcal{S}$ be the set of all multicast sessions. For each session $S_i \in \mathcal{S}$, we use a tuple $S_i = (\mathcal{G}_i, s_i, dem_i)$ to denote it. Here, $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ is a subgraph of graph $\mathcal{G}$, where $\mathcal{V}_i \subseteq \mathcal{V}$ represent the set of datacenters involved in session S_i, $\mathcal{E}_i \subseteq \mathcal{E}$ are all the links inter-connecting them. s_i is the source datacenter, and dem_i is the demand of session i, which is the desired rate value for S_i from soure datacenter s_i to other datacenters in $\mathcal{G}_i$ except s_i.

B. Max-Min Fairness

One of the first to use the term max-min fairness was Bertsekas *et al.* [10], who studied bandwidth allocation among flows in a packet network with given link capacities. Afterwards, Nace *et al.* [15] have provided a new definition of max-min fairness based on lexicographic optimization. As a generic way followed by many, we also use the definition by Nace *et al.*, and state the following definitions in our context.

We denote a session rate vector $f(x) = (f_1(x), f_2(x), ..., f_n(x))$ to be an n-vector whose elements are session rate functions which denote the rate allocated to n multicast sessions.

For each multicast session, consider the session rate function $f_i(x) = \sum_{t_i^j \in T_i} x_i^j$ which is defined on a feasible set F of vectors $x = (x_1^1, ...x_1^{|T_1|}, x_2^1, ...x_2^{|T_2|}, ...x_n^1, ...x_n^{|T_n|})$, where T_i is the set of all the inter-datacenter spanning trees that are constructed for multicast session i, $|T_i|$ is the number of spanning trees for session i, and x_i^j is the rate allocated to session i along its j_{th} spanning tree t_i^j.

Following Nace et al., the session rate vector $f^{'}$ is called lexicographically greater than vector $f^{''}$, $f^{'} \succ f^{''}$, if there exists $j \in \{1, ..., n\}$ such that $f_i^{'} = f_i^{''}$, for all $i \in \{1, ..., j-1\}$ and $f_j^{'} > f_j^{''}$. If $f^{'} \succ f^{''}$ or $f^{'} = f^{''}$ then we write $f^{'} \succeq f^{''}$. As an example, if there are three multicast sessions, the session rate vector $(1, 2, 4)$ should be lexicographically greater than $(1, 1, 5)$. Hence, $(1, 2, 4) \succ (1, 1, 5)$.

The lexicographical maximization problem for given F and session rate vector $f(x)$ is denoted by

$$lexmax_{x \in F} f(x) = (f_1(x), f_2(x), ..., f_n(x)), \tag{1}$$

and consists in finding a vector $x^* \in F$ for which $f(x^*)$ is lexicographically maximal over F, that is, for all $x \in F$, $f(x^*) \succeq f(x)$.

Logically, the lexicographic optimization can be treated as a sequential optimization process where we first maximize $f_1(x)$ on the entire feasible set F, next we maximize $f_2(x)$ on the resulting optimal set, and so on.

To clearly define the max-min fairness optimization problem, we need to introduce another vector. Let $\langle y \rangle = (\langle y \rangle_1, \langle y \rangle_2, ..., \langle y \rangle_m)$ denote a version of vector $y = (y_1, y_2, ..., y_m) \in \mathbb{R}^m$ ordered in the non-decreasing order. Thus, the max-min fairness optimization problem for given F and $f(x)$ is as follows:

$$lexmax_{x \in F} \langle f(x) \rangle \tag{2}$$

The difference between lexicographic optimization and max-min fairness optimization is that the order of sessions in lexicographic optimization case is known. However, in the latter case, the order is unknown beforehand. We first try to maximize the minimum rate that all sessions can get. When some sessions are saturated due to capacity constraints, we get $\langle f(x) \rangle_1$. Then we try to maximize the second minimum rate $\langle f(x) \rangle_2$, and so on.

C. Maximum Concurrent Multi-Tree Multicast

The first step naturally leads us to formulate the maximum concurrent multi-tree multicast problem, which was proposed by Cui *et al.* [8]. The objective is to maximize λ, such that, for each multicast session $S_i \in \mathcal{S}$, $\lambda \cdot dem_i$ units of the respective data can be simultaneously routed, subject to session conservation and inter-datacenter link capacity constraints. λ is the equal maximal fraction of all demands. Using path-flow linear programming formulation, we have

$$\textbf{P1}: maximize \quad \lambda \tag{3}$$

$$subject\ to \quad f_i(x) \geq \lambda \cdot dem_i, \forall i \tag{4}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{|T_i|} x_i^j \cdot \delta_e(t_i^j) \leq c_e, \forall e \in \mathcal{E} \tag{5}$$

$$\lambda \geq 0, x_i^j \geq 0, \forall i, \forall j \tag{6}$$

where $\delta_e(t_i^j)$ is an indicator function. $\delta_e(t_i^j) = 1$ if link e appears in the spanning tree t_i^j. $\delta_e(t_i^j) = 0$, otherwise. **P1** enforces fairness by requiring that the comparative rate of traffic routed for different multicast sessions satisfies the comparative rate of their demands.

D. Max-Min Fair Multi-Tree Multicast

However, the optimal solution of **P1** may significantly reduce network utilization. The reason is that some sessions can easily be saturated due to the limited number of spanning trees. In that situation, the maximization of λ is thus reduced to the maximization of the rate of the session which is congested first, leaving other sessions un-optimized.

The inefficiency of solution **P1** motivates us to take advantage of max-min fairness. Max-min fairness can improve

network utilization as well as potentially increase the throughput for some sessions while maintaining fairness by making each session obtains the maximum possible rate. According to the definition of max-min fairness we presented before, we have the following formulation for the max-min fair multi-tree multicast problem.

$$\mathbf{P2}: lexmax_{x\in F} \quad \langle \boldsymbol{\lambda} \rangle \tag{7}$$

$$subject \quad to \quad f_i(x) \geq \lambda_i \cdot dem_i, \forall i \tag{8}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{|T_i|} x_i^j \cdot \delta_e(t_i^j) \leq c_e, \forall e \in \mathcal{E} \tag{9}$$

$$\lambda_i \geq 0, x_i^j \geq 0, \forall i, \forall j \tag{10}$$

In the formulation above, instead of lexicographically maximizing the non-decreasing session rate vector $\langle f(x) \rangle$, we use a new function $\lambda_i(x) = f_i(x)/dem_i$. Its non-decreasing order vector is denoted by $\langle \boldsymbol{\lambda} \rangle$, which is shown in the objective of the formulation. As in **P1**, the attractive feature is that the absolute value of dem_i becomes meaningless, we can focus on the relative importance of the session. Apparently, we can deal with the value of each demand as a weight. Hereafter, we refer to the rate allocation in our formulation as weighted max-min fairness.

III. Achieving weighted max-min fairness

A. Algorithm for max-min fair muti-tree multicast problem

Traditionally, based on the linear programming formulation, the problem **P1** and **P2** can be solved by using a standard LP solver. However, as the number of spanning trees can be very large, finding the exact solutions by standard LP solvers can be slow and expensive. For a complete graph with n vertices, Cayley's formula gives the number of spanning trees as n^{n-2}. This means, if one of the multicast sessions involving seven datacenters, the number of spanning trees can be 16807 (Cayley's formula [14]). In addition, the number of concurrent multicast sessions can also be very large, which causes the LP solver method ineffective. Instead, we are looking for a fully polynomial time approximation scheme (FPTAS). An FPTAS is an algorithm which takes an instance of an optimization problem and a parameter $\epsilon > 0$ and, in polynomial time, produces a solution that is within a factor $1 - \epsilon$ of being maximal. Its running time is polynomial in the size of the graph ($|\mathcal{V}|$ and $|\mathcal{E}|$), the number of multicast sessions n, and $1/\epsilon$.

The FPTAS algorithm we present in this paper is based on the variable-size increment technique developed by Garg *et al.* [16], which was later improved by Cui *et al.* [8]. Before we presenting our algorithm, we first formulate the LP dual to **P1**.

$$\mathbf{D1}: minimize \quad \sum_{e\in\mathcal{E}} c_e \cdot d_e \tag{11}$$

$$subject\ to \quad \sum_{e\in\mathcal{E}} \delta_e(t_i^j) \cdot d_e \geq l_i, \forall t_i^j \in T_i, \forall i \tag{12}$$

$$\sum_{i=1}^{n} l_i \cdot dem_i \geq 1 \tag{13}$$

$$d_e \geq 0, \forall e \in \mathcal{E}, l_i \geq 0, \forall i \tag{14}$$

The d_e variable holds the link length which is dual to each primal capacity constraint. The l_i variable holds the length of directed minimum spanning tree per each multicast session and is dual to the satisfaction rate λ_i. Hence, we can treat **D1** as the problem of assigning length d_e to each $e \in \mathcal{E}$, such that for session S_i, the length of any spanning tree in T_i is at least l_i, and the sum of l_i by dem_i over all sessions is at least 1.

Algorithm 1: Maximum Concurrent Flow (*MCF*)

```
1  ∀e ∈ E, d_e ← β/c_e
2  x_i^j ← 0, t_i^j ∈ T_i, i = 1, ..., n
3  while Σ_{e∈E} c_e · d_e < 1 do                    /* phase */
4      for i ← 1 to n do                             /* iteration */
5          dem'_i ← dem_i
6          while Σ_{e∈E} c_e · d_e < 1
7          and dem'_i > 0 do                         /* step */
8              t ← directed minimum spanning tree in T_i
               using d_e
9              c ← min{dem'_i, min_{e∈t} c_e}
10             dem'_i ← dem'_i − c
11             f(t) ← f(t) + c
12             ∀e ∈ t, d_e ← (1 + ε c/c_e)
13 ∀i = 1, ..., n, ∀t ∈ T_i, f(t) = f(t) / log_{1+ε} (1+ε)/β
14 ∀i = 1, ..., n, ∀t ∈ T_i, f_i = Σ f(t)
```

The algorithm for the maximum concurrent multi-tree multicast problem (**P1**), henceforth referred to as *MCF*, is shown above. Initially, we assign the length of each link d_e to be β/c_e, where β is a pre-computed value chosen to achieve the desired approximation value ϵ. The algorithm now proceeds in phases. In each phase, there are n iteration. In iteration i, the objective is to ship dem_i units of data from s_i to all receivers inside S_i. This is done in steps, each step calculates the directed minimum spanning tree starting from the source, using the last calculated length variable d_e. For this study, we use the Chu-Liu/Edmond algorithm [17] to find the minimum spanning tree t in a directed graph. The running time of this algorithm is $O(|\mathcal{E}||\mathcal{V}|)$. We then send the remaining demand dem'_i along t. If the remaining demand exceeds the bottleneck capacity on the spanning tree, we only send the traffic equals to its bottleneck capacity. We use $f(t)$ to denote the rate of session allocated on spanning tree t. For every c units data sent over the link e, its length variable d_e is updated by a factor of $1 + \epsilon\frac{c}{c_e}$. The entire procedure stops as soon as the objective function value of **D1** is at least one: $\sum_{e\in\mathcal{E}} c_e \cdot d_e > 1$.

Because the number of times each session flow increased is $\log_{1+\epsilon}\frac{1+n\epsilon}{\beta}$, we scale down each session flow by a factor of $\log_{1+\epsilon}\frac{1+n\epsilon}{\beta}$. Hence we can get a feasible session rate allocation.

The solution of **P1** gives us the equal maximal fraction of all demands constrained by link capacities. However, as we noted before, this solution can lead to the inter-datacenter network runs at a very low utilization. We have developed a new algorithm that considers both the inter-datacenter network efficiency and fairness among multicast sessions. The detailed procedure of the algorithm is given in Algorithm 2.

Algorithm 2: Max-min Fair Multi-Tree Multicast (*Blossom*)

1 $\forall e \in \mathcal{E}, d_e \leftarrow \beta/c_e$
2 $\Gamma = \mathcal{S}$
3 **while** $\Gamma \neq NULL$ **do** /* stage */
4 **while** $curDL - lastDL < 1$ **do** /* phase */
5 **for** $i \leftarrow 1$ **to** n **do** /* iteration */
6 $dem'_i \leftarrow dem_i$
7 $flag = False$
8 **while** $dem'_i > 0$ **do** /* step */
9 **if** $flag == True$ **then**
10 break
11 $t \leftarrow$ directed minimum spanning tree in T_i using d_e
12 **for** $e \in t$ **do**
13 **if** $d_e \geq 1/c_e$ **then**
14 $\Gamma = \Gamma \backslash S_i$
15 $flag = True$
16 break
17 $c \leftarrow min\{dem'_i, min_{e \in t} c_e\}$
18 $dem'_i \leftarrow dem'_i - c$
19 $f(t) \leftarrow f(t) + c$
20 $\forall e \in t, d_e \leftarrow (1 + \epsilon\frac{c}{c_e})$
21 $curDL = CalcuateD(l)$
22 $lastDL = curDL$
23 $\forall i = 1, ..., n, \forall t \in T_i, f(t) = \frac{f(t)}{\log_{1+\epsilon}\frac{1+\epsilon}{\beta}}$
24 $\forall i = 1, ..., n, \forall t \in T_i, f(T_i) = \sum f(t), \lambda_i = \frac{f(T_i)}{dem_i}$

We refer to the algorithm for the max-min fair multi-tree multicast problem as *Blossom*. The gist of *Blossom* is a saturation test shown between line 12 and line 16. During each iteration, we examine if the demand of session S_i cannot by any means be further increased. To do this, for each link on the minimum spanning tree of S_i, we compare each current length value d_e with $1/c_e$. The link e is saturated as soon as $d_e \geq 1/c_e$. It should be noted that, in *MCF*, the algorithm terminates when $\sum_{e \in \mathcal{E}} c_e \cdot d_e > 1$. To further improve the network utilization, for those sessions who can not pass the saturation test, we remove them from the session set $\mathcal{S}$. In *Blossom*, we continue to allocate session rates to other sessions. For this purpose, we change the termination condition (see line 4). In each iteration, we evaluate the incremental value of $\sum_{e \in \mathcal{E}} c_e \cdot d_e$ since the last iteration. Thus, at least one session is saturated and removed from $\mathcal{S}$. When all sessions are removed, the algorithm is terminated.

B. Algorithm correctness and complexity

Following the same approach as Grag *et al*., we can prove the following sequence of lemmas and theorems. Let $\alpha(d) \stackrel{\text{def}}{=} \sum_{i=1}^{n} mst_i(d)$ be the sum of minimum spanning tree length of all multicast sessions. Let $\langle \boldsymbol{\lambda}^* \rangle = lexmax_{x \in F} \langle \boldsymbol{\lambda} \rangle$ be the result returned by *Blossom*.

Lemma 1. Linear program **D1** is equivalent to finding a length function $d : \mathcal{E} \rightarrow \mathbb{R}^+$ such that $\frac{D(d)}{\alpha(d)}$ is minimized.

Lemma 2. Scaling the final session flow by $\log_{1+\epsilon}\frac{1+\epsilon}{\beta}$ (line 23) yields a feasible primal solution of value $\langle \boldsymbol{\lambda}^* \rangle = (\langle \lambda^* \rangle_i) > (\frac{P_i - 1}{log_{1+\epsilon}\frac{1+\epsilon}{\beta}})$, where P_i is the total number of steps when S_i is removed from the session set $\mathcal{S}$.

Lemma 3. The final flow scaled by $log_{1+\epsilon} 1/\beta$ has a value at least $(1 - 3\epsilon)$ times ζ_1, when $\beta = (|\mathcal{E}|/(1-\epsilon))^{-1/\epsilon}$. ζ_1 is the value of $\frac{D(d)}{\alpha(d)}$ after the first stage of *Blossom* is finished.

Lemma 4. If $\beta = (|\mathcal{E}|/(1-\epsilon))^{-1/\epsilon}$, *Blossom* terminates after at most $P_n = 1 + \frac{\zeta_1}{\epsilon} log_{1+\epsilon}\frac{|\mathcal{E}|}{1-\epsilon}$ steps.

Theorem 1. The running time of *Blossom* is $O(\epsilon^{-2} log|\mathcal{E}|(2n^2 logn|\mathcal{E}||\mathcal{V}| + n|\mathcal{E}|^2|\mathcal{V}|))$.

The *Blossom* algorithm also gives us the following theorem, we can prove it by introduction.

Theorem 2. The resulted satisfaction rate vector $\langle \boldsymbol{\lambda}^* \rangle$ is indeed weighted max-min fair.

IV. Evaluation

We use simulation to evaluate the efficiency of *Blossom* compared to the *MCF* algorithm which is using a proportional fairness criterion. We leave the implementation of *Blossom* and assessing its performance on a real cloud platform to future work.

TABLE I
Available bandwidth (Mbps) for micro instances between EC2 datacenters. VA, CA, EU, SG, BR, OR and JP correspond to Virginia, California, Ireland, Singapore, Brazil, Oregon and Tokyo, respectively

	VA	CA	EU	SG	BR	OR	JP
VA		24	35.4	16	26	19.1	18.2
CA	31.1		24.3	31.4	19	69.7	43,7
EU	110	9.65		21.7	13.8	14.6	7.13
SG	11.4	27.9	16		9.42	15	20.8
BR	29	15.4	16.8	13.8		18.4	4.24
OR	25.4	85.4	25.5	11.8	13.1		31.4
JP	31	35.9	16.9	12.4	3.85	54.4	

We measure the available bandwidth between 7 Amazon EC2 datacenters using `Iperf` [18] as an input of inter-datacenter link capacity constraints for both algorithms, and report them in Table I. 10 multicast sessions are randomly created to simulate 10 pieces of data that are needed to be delivered at the same time. The session demand and the number of datacenters involved in each session are generated uniformly at random in the range of [1,10] and [2,7], respectively. The source datacenter also is randomly selected.

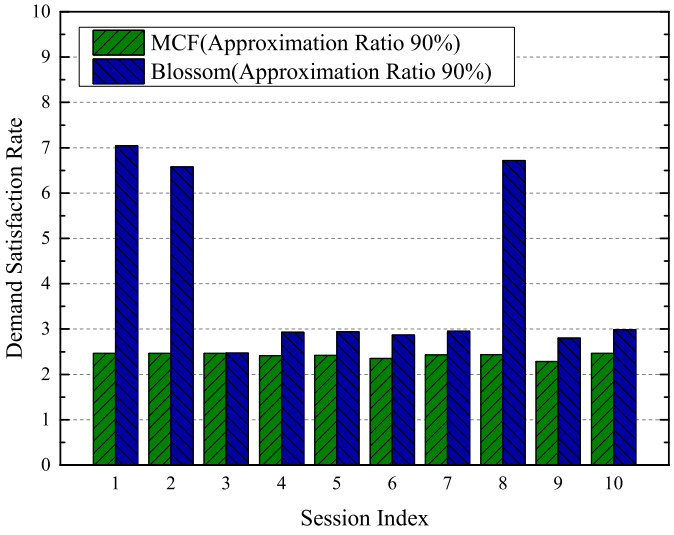

Fig. 1. Demand satisfaction rates in 10 multicast sessions.

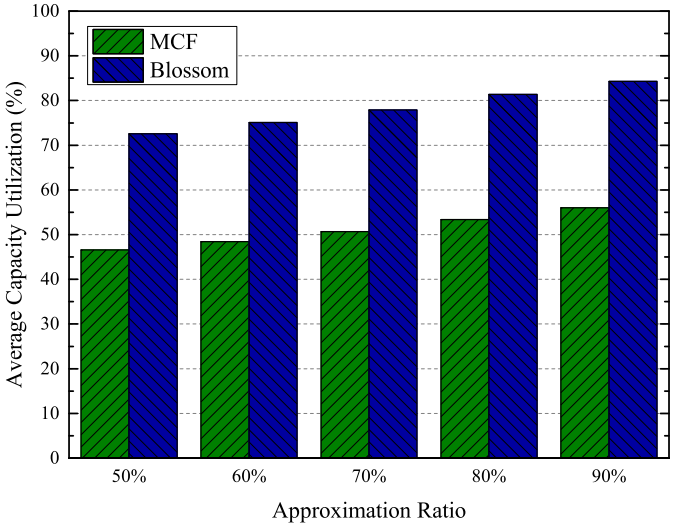

Fig. 2. Average capacity utilization in the inter-datacenter network.

Fig. 1 compares demand satisfaction rates in 10 multicast sessions with approximation ratio 90% for *MCF* and *Blossom*. We can find session 3 has nearly the same rate by using different algorithms. A possible explanation for this might be that this session is saturated during the first stage of our algorithm. While after it is saturated, we continue to increase other sessions' rates until all sessions are saturated. In the best case, *Blossom* with 90% approximation ratio achieves 1.86 times higher demand satisfaction rate than *MCF* for session 1. This figure also shows that our *Blossom* algorithm is indeed max-min fair, because it maximizes the minimum share of a multicast session whose demand is not fully satisified.

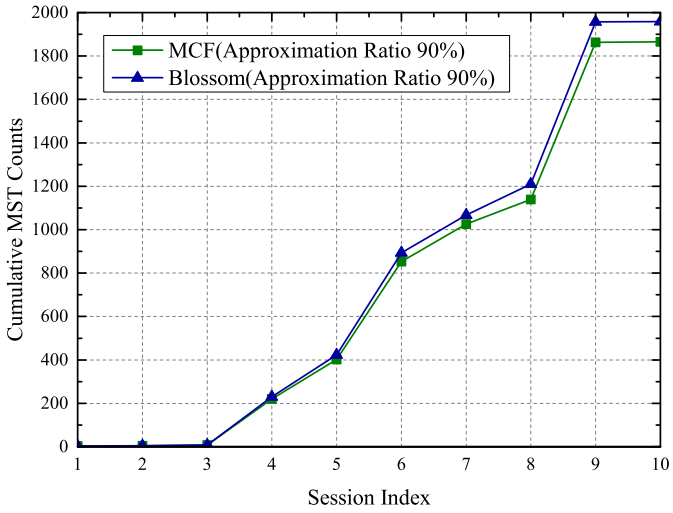

Fig. 3. Number of minimum spanning trees used.

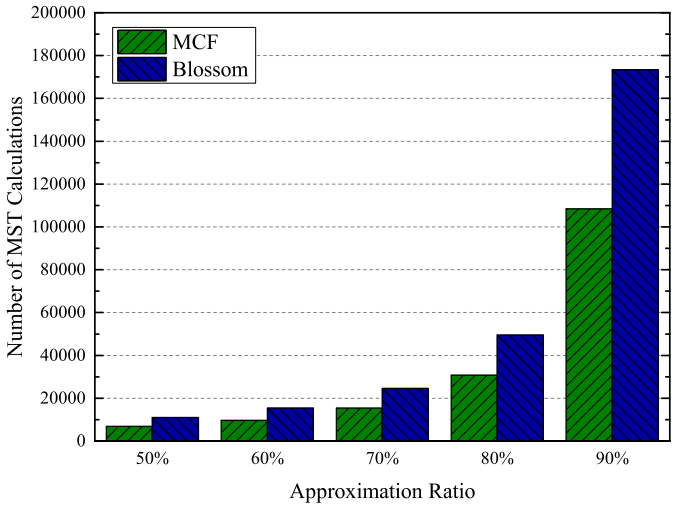

Fig. 4. Comparison of runtime (minimum spanning tree calculations).

Fig. 2 shows the average capacity utilization of all inter-datacener links. We consider the average capacity utilization as a ratio between the allocated bandwidth and the link capacity. Comparing the two results, it can be seen that *Blossom* has at least 50.4% higher capacity utilization than *MCF* irrespective of the approximation ratio. The capacity utilization is increased to 56.0% and 84.3% in *MCF* and *Blossom* respectively when we increase the approximation ratio to 90%.

The reason that why *Blossom* outperforms *MCF* can be found in Fig. 3. With *Blossom*, we explore much more spanning trees for each session than with *MCF*. In the first

three sessions, the number of spanning trees can be used is limited by the small session sizes, both *Blossom* and *MCF* use 3, 2 and 3 spanning trees. When the session sizes are growth, the differences between the number of spanning trees used in the two algorithms become remarkable. For example, in session 4, *Blossom* uses 10 more spanning trees than *MCF*. As a result, much more available bandwidth of the inter-datacenter network are leveraged. Thus, the capacity utilization is increased.

Blossom increases the rates of multicast sessions as well as maximizes the capacity utilization of inter-datacenter links, but at the cost of increasing the runtime of bandwidth allocation algorithm. The runtime of the two algorithms is shown in Fig. 4. Both *Blossom* and *MCF* encountered much more minimum spanning tree calculations as the value of approximation ratio increases. In particular, by increasing the approximation ratio from 80% to 90%, the number of minimum spanning trees calculated by each algorithm is increased by a factor of 2.5.

V. Related Work

Many recent works perform centralized traffic engineering to improve capacity utilization of inter-datacenter networks [6], [7]. The core of these approaches is a max-min fair multicommodity flow solution. However, when each commodity becomes a multicast session which consists of a source datacenter and several destination datacenters, the problem becomes challenging even in a centralized way.

Airlift [19] aggregates the multicast session originated from the same datacenter to the same set of destination datacenters as an aggregated session. It achieves the maximal throughput while at the cost of sacrificing the fairness among sessions involved in the same aggregated session.

Some papers in peer-to-peer research area also consider the problem of the tradeoff between capacity utilization and multicast session fairness [20], [21], [22]. In peer-to-peer networks, distributed algorithms are desired because individual peer doesn't have a global view on network topology. *Blossom* executes bandwidth allocation in a centralized fashion.

VI. Conclusion

In this paper, we developed a bandwidth allocation algorithm named *Blossom* for cloud service providers to distribute contents among datacenters. Our main contribution is an approximation algorithm that achieves max-min fair bandwidth allocation for competing multicast sessions in polynomial time. We carried out trace-driven simulation to evaluate the performance of *Blossom*, and showed that the algorithm can achieve high capacity utilization while maintaining inter-session fairness. Currently, We are implementing a session controller based on POX [23], which can obtain the bandwidth demand and available bandwidth sent by the participants of multicast sessions. Additional future work opportunities include the design of a rate limiting mechanism in user space to flexibly allocate bandwidth.

Acknowledgment

This paper was partly written during the first author visit the University of Toronto and professor Baochun Li in 2014. His encouragement greatly improved this paper. The first author would also like to acknowledge the financial support from the program of China Scholarship Council.

References

[1] "Google - data centers," http://www.google.com/about/datacenters.
[2] "Amazon - global infrastructure," https://aws.amazon.com/about-aws/globalinfrastructure/.
[3] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, Dec. 2008.
[4] Z. Wu and H. V. Madhyastha, "Understanding the latency benefits of multi-cloud webservice deployments," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 1, pp. 13–20, 2013.
[5] V. K. Adhikari, S. Jain, and Z.-L. Zhang, "Youtube traffic dynamics and its interplay with a tier-1 ISP: an ISP perspective," in *Proc. ACM IMC*, 2010, pp. 431–443.
[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 *Proc. ACM SIGCOMM*, 2013, pp. 15–26.
[7] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *Proc. ACM SIGCOMM*, 2013, pp. 3–14.
[8] Y. Cui, B. Li, and K. Nahrstedt, "On achieving optimized capacity utilization in application overlay networks with multiple competing sessions," in *Proc. ACM SPAA*, 2004, pp. 160–169.
[9] X. Zheng, C. Cho, and Y. Xia, "Optimal peer-to-peer technique for massive content distribution," in *Proc. IEEE INFOCOM*, April 2008.
[10] D. P. Bertsekas and R. G. Gallager, *Data Networks*. Prentice-Hall, Inc., 1992, vol. 2.
[11] E. Danna, S. Mandal, and A. Singh, "A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering," in *Proc. IEEE INFOCOM*, 2012, pp. 846–854.
[12] A. Shieh, S. Kandula, A. Greenberg, and C. Kim, "Seawall: performance isolation for cloud datacenter networks," in *Proc. USENIX HotCloud*, 2010, pp. 1–1.
[13] "CLP: Coin-or linear programming," https://projects.coin-or.org/Clp.
[14] A. Cayley, "A theorem on trees," *Quart. J. Math*, vol. 23, no. 376-378, p. 69, 1889.
[15] D. Nace and M. Pióro, "Max-min fairness and its applications to routing and load-balancing in communication networks: A tutorial," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 4, pp. 5–17, 2008.
[16] N. Garg and J. Koenemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM Journal on Computing*, vol. 37, no. 2, pp. 630–652, 2007.
[17] Y.-J. Chu and T.-H. Liu, "On shortest arborescence of a directed graph," *Scientia Sinica*, vol. 14, no. 10, p. 1396, 1965.
[18] "Iperf - the tcp/udp bandwidth measurement tool," http://iperf.fr/.
[19] Y. Feng, B. Li, and B. Li, "Airlift: Video conferencing as a cloud service using inter-datacenter networks," in *Proc. IEEE ICNP*, 2012, pp. 1–11.
[20] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments," in *Proc. ACM SOSP*, 2003, pp. 298–313.
[21] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, "Utility maximization in peer-to-peer systems," in *Proc. ACM SIGMETRICS*, 2008, pp. 169–180.
[22] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *Proc. IEEE ICNP*, 2006, pp. 2–11.
[23] "About POX," http://www.noxrepo.org/pox/about-pox/.