Calantha: Content Distribution across Geo-Distributed Datacenters

Yangyang Li, Linchao Zhang, Yue Jia, Yong Liao, Haiyong Xie
Innovation Center
China Academy of Electronics and Information Technology, Beijing, 100041, China
Email: {yli, lzhang, yjia, yliao, hxie}@csdslab.net

***Abstract*—Large cloud service providers often replicate data to multiple geographically distributed datacenters for availability and service quality purposes. The enormous amount of data needed to be shuffled among datacenters call for efficient schemes to maximally exploit the capacity of the inter-datacenter networks. In this paper, we propose *Calantha*, a new rate allocation scheme that improves the reliability and operability of content distribution across geo-distributed datacenters, without sacrificing capacity utilization and max-min fairness among competing sessions. *Calantha* leverages hop-constrained spanning tree to enhance the reliability of inter-datacenter links. A novel approximation algorithm is proposed to solve the rate allocation problem in polynomial time and achieve α-optimal approximation. Our simulation results have shown that we can reduce the number of spanning trees by 44.5%, as well as has 2.7% more average capacity utilization and 1.0% less minimum spanning tree calculations.**

I. Introduction

For the purpose of mitigating latency and increasing service availability, cloud service providers build geographically distributed networks of datacenters around the world and replicate data across multiple geographic locations [1], [2]. Hence, maximally exploiting the capacity of inter-datacenter networks is essential for cloud service providers to efficiently distribute the exponential-growing content among their datacenters. Software-driven WAN (SWAN) presented in [2] is such an example, where centralized traffic engineering was proposed to boost the utilization of inter-datacenter networks by taking the priority requirements of different traffic categories into account.

The essence of the traffic engineering proposed by SWAN is enabling multiple concurrent route updates in short time scales. Not only Microsoft, Google adopts a similar centralized traffic engineering mechanism in its B4 [1] architecture that splits traffic flows among multiple inter-datacenter paths to balance capacity against application priority. By employing centralized traffic engineering, utilization can be improved by near 100%. However, under this circumstance, the bandwidth efficiency can still be quite low because transmitting data replicas via unicast is not quite suitable for one-to-many communications.

In reality, one-to-many communications are quite commonly applied within inter-datacenter networks for bulk data transferring. Bulk data often needs to be replicated to the datacenters close to users. Typical services across geo-distributed datacenters are content distribution, data migration, disaster recovery and data backup. Those services could improve the bandwidth efficiency of inter-datacenter links by using multicast techniques. For example, *Airlift* [3] aggregates video conferences to a small number of multicast sessions among datacenters. However, it achieves the maximal throughput at the cost of sacrificing the fairness among sessions involved in the same aggregated session. Afterward, multicast through multiple spanning trees, named *Blossom* [4], was proposed to improve the throughput of multicast sessions while maintaining fairness among multiple multicast sessions.

Similar to B4, *Blossom* splits traffics of multicast sessions to multiple inter-datacenter spanning trees. Theoretically, it could improve the throughput of multicast sessions as well as promote the utilization of inter-datacenter links. However, implementing *Blossom* in reality faces a few challenges. On one hand, the number of spanning trees used by the multicast session grows exponentially as session scale increases. It is common that cloud service providers use more than five datacenters to replicate data [5], which means at most 125 spanning tress shall be used to boost the content distribution. Nevertheless, implementing one deployable multipath protocol is already tough and hardware consuming [6]. Secondly, content distribution through a large number of hops can cause potential reliability issue for content distribution, because each link can fail independently so as to cause an entire path to fail. An early study [7] pointed out that the data paths should be constrained to 3 hops, so as to avoid performance degradation of the inter-datacenter network.

In order to overcome the drawbacks of existing work such as *Blossom* and B4, we intend to limit hops which can be applied by inter-datacenter spanning trees. In one of our simulation experiments, we apply *Blossom* algorithm with 7 datacenters and 10 multicast sessions, 459 spanning trees were used. The number of spanning trees can be reduced to 214 and 226 if we constrained the largest hops in the spanning trees to 2 hops and 3 hops, respectively. Reducing the used spanning trees can greatly reduce the complexity of content distribution system. Interestingly, we found that even if we constrained the spanning trees that can be used by each multicast session, the rate of such multicast sessions did not have obvious degradation. For instance, in the same simulation experiment, one of the session rates is 11.02 Mbps with *Blossom*, and it changes to 11.74 Mbps and 10.15 Mbps respectively with 2 hops constrained and 3 hops constrained spanning trees.

In this paper, we present *Calantha*, a content distribution

978-1-5386-2784-6/17/$31.00 ©2017 IEEE

method that uses hop-constrained spanning trees to enhance the reliability of inter-datacenter links as well as boost the utilization of inter-datacenter links. Moreover, our method can also achieve fairness among different sessions. *Calantha* is formulated as a variant of the maximum concurrent flow problem [8]. A fully polynomial time approximation scheme (FPTAS) is developed to provide a faster and simpler solution to the problem. The key idea towards making the problem tractable is to formulate the hop-constrained minimum spanning tree problem to multi-commodity flow problem. Hence, existing heuristic algorithm can be used to make *Calantha* achieve α-optimal approximation in polynomial time. Through extensive trace-driven simulation study, we show that our approach is substantially more efficient than those state-of-the-art approaches previously proposed in the literature.

The remainder of this paper is organized as follows. We introduce the motivation in Sec. II and then formulate the rate allocation problem in Sec. III. In Sec. IV, we briefly describe a new algorithm and discuss the time complexity of the proposed algorithm. Amazon EC2 trace-based simulation results are presented in Sec. V, followed by related work in Sec. VI and our conclusion in Sec. VII.

II. Motivation

We use a toy example to illustrate the basic idea of *Calantha*. Fig. 1 depicts an inter-datacenter network with five datacenters connected by directed links, where the capacity of each link at each direction is 1 unit. We have two multicast sessions, both of which need to deliver data to other datacenters from D_1 at the rate of 1 unit. Session 1 has only one destination, D_2. Session 2 has three destinations, D_2, D_3, and D_4.

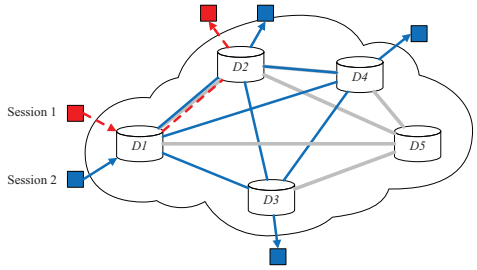


Fig. 1. A toy example of rate allocation for two multicast sessions in an inter-datacenter network.

Suppose that the goal is to maximize the capacity utilization of all links. One could allocate all the available capacity of link $\{D_1 \rightarrow D_2\}$ to session 1 or session 2. However, this can lead one of these sessions to be starved. One attempt to solve this problem was to introduce fairness among competing sessions. More specifically, the demands of both sessions are required to be equally satisfied and the strategy is to find spanning trees that maximize session rate in equal portions per demand. As a result, the optimal rate allocation for the two sessions should both be 0.6 unit. session 1 uses link $\{D_1 \rightarrow D_2\}$ at a rate of 0.6 unit, session 2 uses spanning trees $\{D_1 \rightarrow D_3 \rightarrow D_2, D_1 \rightarrow D_4\}$ and $\{D_1 \rightarrow D_2, D_1 \rightarrow D_3, D_1 \rightarrow D_4\}$ at the rate of 0.2 unit and 0.4 unit respectively.

Obviously, this solution causes the inter-datacenter links to be under-utilized, since the rate of session 2 could be increased by using other unsaturated links. It is preferable to increase the rate of session 2 while not decreasing the rate of session 1, as the links other than $\{D_1 \rightarrow D_2\}$ are not shared with session 1. The *Blossom* algorithm proposed in [4] computes rate allocation and spanning trees to maximize the capacity utilization and adhere to the max-min fairness criterion [9]. The resulted rate allocation are 0.6 unit for session 1, and 1.2 units for session 2.

The outcome of *Blossom* in solving the example shown in Fig. 1 is quite attractive. However, *Blossom* requires session 2 to use 8 spanning trees. Furthermore, two of the spanning trees have 3 hops, e.g., $\{D_1 \rightarrow D_4 \rightarrow D_3 \rightarrow D_2\}$. Large number of spanning trees leads to higher management cost of maintaining those trees; deep trees often mean more resources are needed to transfer the data to its destination datacenters. These two reasons motivate us to explore the design space of reducing the number of spanning trees used by each session as well as making the spanning trees use less hops.

Our basic idea is to use hop-constrained minimum spanning trees instead of minimum spanning trees to enhance the reliability of inter-datacenter links. The rate allocation in the example shown in Fig. 1 is 0.6 unit for session 1, and 1.2 units for session 2, the same as *Blossom*. What's interesting here is that session 2 only needs to using 3 spanning trees, $\{D_1 \rightarrow D_3, D_1 \rightarrow D_4 \rightarrow D_2\}$, $\{D_1 \rightarrow D_4 \rightarrow D_2, D_1 \rightarrow D_4 \rightarrow D_3\}$, and $\{D_1 \rightarrow D_2, D_1 \rightarrow D_3, D_1 \rightarrow D_4\}$. Each of the spanning trees is allocated 0.4 unit, and the deepest spanning tree has 2 hops only.

III. Model and Formulation

We consider a scenario where a cloud service provider runs multiple datacenters. These datacenters are fully connected with each other, and the inter-datacenter network is a complete graph denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of datacenters, and $\mathcal{E} = \{(m, n) : m, n \in \mathcal{V}\}$ is the set of directed links between datacenters. For a directed link $e = (m, n) \in \mathcal{E}$, we use c_{mn} to denote the capacity of link e, which is the maximum rate of data transmission over e. To be mentioned here, we assume that fixed capacity of each link is reserved for content distribution, and the rate allocation is executed periodically.

Let $\mathcal{S}$ be the set of all multicast sessions among datacenters to distribute contents. Each session $S_i \in \mathcal{S}$ is represented as a tuple $S_i = (\mathcal{G}_i, s_i, dem_i)$, where $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ is a subgraph of graph $\mathcal{G}$. $\mathcal{V}_i \subseteq \mathcal{V}$ represents the set of datacenters involved in session S_i, and $\mathcal{E}_i \subseteq \mathcal{E}$ are all the links connecting them. s_i is the root of the multicast tree (i.e., the source datacenter), and dem_i is the desired content distribution rate from source datacenter s_i to other datacenters in $\mathcal{G}_i$. Suppose that there are $|S|$ multicast sessions for content distribution. For each multicast session S_i, let T_i be the set of inter-datacenter spanning trees that are used by session S_i to

distribute contents, and x_i^j be the allocated rate of session S_i's jth spanning tree.

To constrain the number of hops for each path within a given integer value H (from the source datacenter s_i to every other datacenter), as well as to limit the total number of spanning trees in T_i, we formulate the hop-constrained minimum spanning tree problem to a multi-commodity flow problem as the following.

$$\text{minimize} \left(- \sum_{(m,n)\in\mathcal{E}_i} c_{mn} \cdot x_{mn} \right) \tag{1}$$

$$\text{subject to} \sum_{m\in\mathcal{V}_i} x_{mn} = 1, \forall n \in \mathcal{V}_i \setminus \{s_i\}, \tag{2}$$

$$\sum_{m\in\mathcal{V}_i} y_{mn}^k - \sum_{m\in\mathcal{V}_i\setminus\{s_i\}} y_{nm}^k = 0, \tag{3}$$
$$\forall k, n \in \mathcal{V}_i \setminus \{s_i\}, n \neq k,$$

$$\sum_{m\in\mathcal{V}_i} y_{mn}^n = 1, \forall n \in \mathcal{V}_i \setminus \{s_i\}, \tag{4}$$

$$\sum_{(m,n)\in\mathcal{E}_i} y_{mn}^k \leq H, \forall k \in \mathcal{V}_i \setminus \{s_i\}, \tag{5}$$

$$y_{mn}^k \leq x_{ij}, \forall (m,n) \in \mathcal{E}_i, \forall k \in \mathcal{V}_i \setminus \{s_i\}, \tag{6}$$

$$x_{mn} \in \{0,1\}, \forall (m,n) \in \mathcal{E}_i, \tag{7}$$

$$y_{mn}^k \in \{0,1\}, \forall (m,n) \in \mathcal{E}_i, \forall k \in \mathcal{V}_i \setminus \{s_i\} \tag{8}$$

The decision variable x_{mn} [1] represents whether the inter-datacenter link (m,n) is in session S_i's spanning tree, and y_{mn}^k represents whether link (m,n) is in the path from node s_i to node k. Equation (1) represents the cost function to be minimized. Here we choose the residual capacity of each link as cost measurements, hence the minimum spanning tree represents a tree that has the largest idle paths. Equation (2) guarantees that every datacenter in the spanning tree has only one edge entering it. Constraints (3) refers to the flow conservation constraints. Equation (4) and (5) ensure that the hop constraint is guaranteed. Equation (6) states that an inter-datacenter link (m,n), within the spanning tree, can only be one of the paths between the root datacenter s_i and one datacenter k.

As inter-datacenter links are shared by spanning tress from multiple multicast sessions, it is desired to obtain an optimal rate allocation method without exceeding link capacities. The objective is to find a rate allocation plan so that the minimum session rate in the session rate vector had been maximized. Such an objective can be rigorously formulated as a lexicographical maximization problem as the following.

Definition 1. The *rate allocation method* for all $|S|$ sessions is denoted as a vector $\bar{x} = (x_1^1, ...x_1^{|T_1|}, ...x_{|S|}^1, ...x_{|S|}^{|T_n|})$. Under rate allocation method $\bar{x}$, we define the *session rate vector* $f(\bar{x})$ as $f(\bar{x}) = (f_1, f_2, ..., f_{|S|})$, where $f_i = \sum_{t_i^j\in T_i} x_i^j$ is referred to the *session rate* of S_i.

Definition 2. A session rate vector $f^{'}$ is called lexicographically greater than vector $f^{''}$, $f^{'} \succ f^{''}$, if there is $j \in \{1, ..., |S|\}$ 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 $f^{'} \succeq f^{''}$.

For example, if there are three multicast sessions, the session rate vector $(1,2,4)$ is lexicographically greater than $(1,1,5)$. Hence, $(1,2,4) \succ (1,1,5)$.

Definition 3. The lexicographical maximization problem for a given feasible set F and a session rate vector $f(x)$ is denoted as

$$lexmax_{x\in F} f(x) = (f_1(x), f_2(x), ..., f_{|S|}(x)). \tag{9}$$

Solving this problem is to find a vector $x^* \in F$ for which $f(x^*)$ is lexicographically maximal over F, i.e., for all $x \in F$, $f(x^*) \succeq f(x)$.

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

Definition 4. Let $\langle y \rangle = (\langle y \rangle_1, \langle y \rangle_2, ..., \langle y \rangle_t)$ denote a version of vector $y = (y_1, y_2, ..., y_t) \in \mathbb{R}^t$ ordered in the non-decreasing order. The max-min fairness optimization problem for given F and $f(x)$ is as following:

$$\text{lexmax}_{x\in F} \langle f(x) \rangle \tag{10}$$

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. First we try to maximize the minimum rate that can be allocated to all the sessions. Due to capacity constraints, some sessions are saturated, so we have $\langle f(x) \rangle_1$. Then we try to maximize the second minimum rate $\langle f(x) \rangle_2$.

Considering the link capacity constraint, this max-min fairness rate allocation problem can be formulated as following:

$$\text{lexmax} \quad \langle f(x) \rangle \tag{11}$$

$$\text{subject to} \sum_{i=1}^{|S|} \sum_{j=1}^{|T_i|} x_i^j \cdot \delta_e(t_i^j) \leq c_e, \forall e \in \mathcal{E} \tag{12}$$

$$x_i^j \geq 0, \forall i, \forall j \tag{13}$$

Here $\delta_e(t_i^j)$ 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.

IV. Algorithm

The problem defined in (11) $\sim$ (13) can be solved by extending the algorithm proposed in [4]. The algorithm proceeds in phases. In each phase, we first calculate the hop-constrained minimum spanning tree for each session, then we maximize the rates across all multicast sessions and allocate rates on corresponding hop-constrained minimum spanning tree to the

[1] To clarify, we reuse the symbol x here. However, it is clear for the reader to distinguish the meaning from the context. x_i^j represents the rate allocated to session S_i on the jth spanning tree.

TABLE I
AVAILABLE BANDWIDTH (MBPS) ACROSS GEO-DISTRIBUTED AMAZON EC2 DATACENTERS. VA, OR, CA, EU, SG, JP AND BR CORRESPOND TO VIRGINIA, OREGON, CALIFORNIA, IRELAND, SINGAPORE, TOKYO AND BRAZIL, RESPECTIVELY

	VA	OR	CA	EU	SG	JP	BR
VA		65	68.4	54.3	26.9	38	65.7
OR	91.8		72	17.9	20.7	19	26.8
CA	30.1	44.7		16.8	51	20.9	20.8
EU	60.3	40.5	38.4		16.9	26.7	10.9
SG	22.5	33.3	34.6	26.3		59.8	12.4
JP	50.1	52.4	60.1	26.3	57.9		22.1
BR	38.6	24.6	52.5	19.5	8.55	15.3	

sessions that are saturated at the current phase. These sessions whose rates are already maximized are then excluded from the next phrases. The entire procedure stops when all sessions are excluded or all links are saturated. Since this algorithm is inspired by our previous *Blossom* algorithm, and it yields better rate allocations, we name it *Calantha*. *Calantha* means beautiful *Blossom* in Greek.

The essence of *Calantha* is a hop-constrained minimum spanning tree algorithm instead of the minimum spanning algorithm used in *Blossom*. It has been proven that the hop-constrained minimum spanning tree problem is NP-hard [10], [11] and heuristic algorithms proposed in literature, such as [12], [13], can be adopted to solve this problem with manageable cost. In most scenarios, the number of datacenters involved in a multicast session is limited, e.g., Google reported that it has 15 datacenters [14], the decision variables in our hop-constrained minimum spanning tree problem will not be too large. We use the Linear Programming solver in the SCIP Optimization Suite [15] to solve the problem in acceptable running time.

In *Blossom*, we use the Chu-Liu/Edmond algorithm [16] to find the minimum spanning tree in a directed graph. The running time of this algorithm is $O(|\mathcal{E}||\mathcal{V}|)$. *Calantha* uses heuristic algorithm to develop an approximation solution to the hop-constrained minimum spanning tree problem. According to [17], an $O(log|\mathcal{V}|)$-approximation algorithm can be solved with running time $O(|\mathcal{V}|^5 H)$, which ensures that *Calantha* can run in polynomial time. In particular, the running time of *Calantha* is $O((1-\alpha)^{-2} log|\mathcal{E}|(2|S|^2 log|S||\mathcal{V}|^5 H + |S||\mathcal{E}||\mathcal{V}|^5 H))$. More details can be found in our technical report [18].

V. EVALUATION

In this section, we evaluate *Calantha* via extensive simulations and compare it performance with *Blossom* [4] and *MCF* [19]. *MCF* is a variant of the multiple concurrent flow method. We begin with a comparison of number of spanning trees that are used by each session with different algorithms. Then we verify whether our method still achieves high average capacity utilization of inter-datacenter links, and whether the rate allocation among competing multicast session is still max-min fair. Finally, we evaluate the running time of *Calantha* with various approximation parameters.

TABLE II
5 SESSIONS WITH DIFFERENT DEMANDS AND DESTINATION SETS

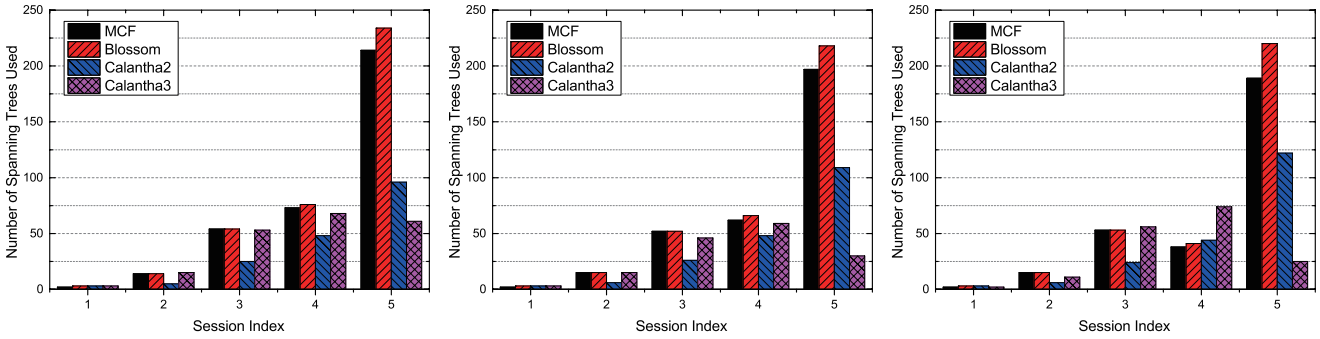
$SessionID$	$Source$	$DestinationSet$	$Demand$
1	1	[7,6]	1
2	3	[5,1,6]	5
3	6	[1,7,3,5]	7
4	4	[1,7,6,5,3]	2
5	2	[1,7,6,5,4,3]	3

To simulate the inter-datacenter link capacity constraints of a real environment, the parameters used in our simulation are based on our measurement of the available bandwidth between 7 Amazon EC2 datacenters. The available bandwidth is measured via `Iperf` [20], and we report our measurement results in Table I. We conduct five groups of simulations in a 5 mutlticast sessions and 7 datacenters involved inter-datacenter network, with the approximation parameter α set as 50%, 60%, 70%, 80%, and 90%. We randomly set the demand rate of each session in the range of [1,10], and randomly set the number of datacenters in each session in the range of [2,7]. The source datacenter is also randomly selected and it has 5 pieces of data to be distributed at the same time. The session information is listed as in Table. II, and the numbers in the range of [1,7] are used to denote different datacenters.

Fig. 2 shows the number of spanning trees used in each session by *MCF*, *Blossom*, *Calantha2* (2-hop constrained), and *Calantha3* (3-hop constrained). With all approximation parameters, *Blossom* uses the largest number of spanning tress, while *Calantha* uses the least. In particular, *Calantha2* allocates less than half of the spanning trees that are used in session 2, 3, and 5 compared with both *Blossom* and *MCF*. We also find that when the destination set in the sessions is not large, such as session 1, the number of spanning trees yielded by different algorithms does not have noticeable differences. As the destination set grows larger, the differences become more significant. For example, when $\alpha = 80\%$, in session 5, the number of spanning trees used by *MCF*, *Blossom*, *Calanth2* and *Calantha3* is 189, 220, 122 and 25, respectively, where *Calanth2* and *Calantha3* yield 44.5% and 88.6% less trees as compared to *Blossom*, respectively.

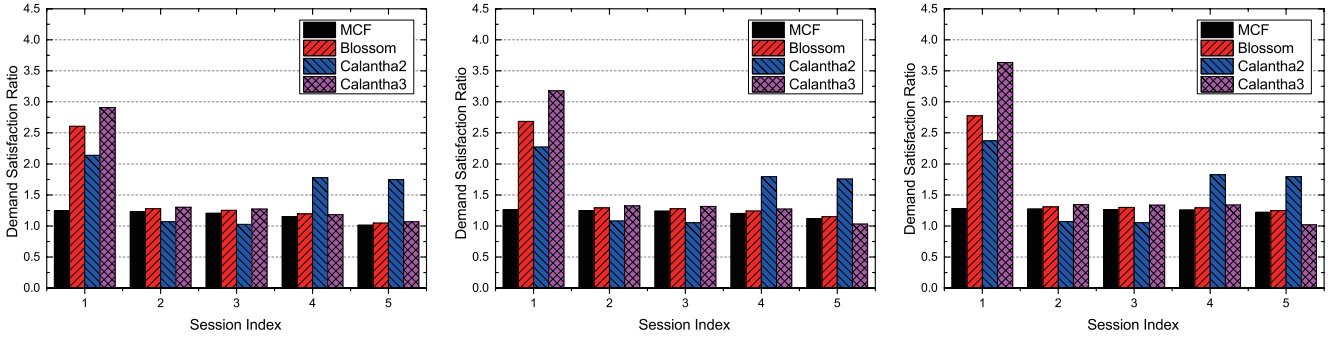
Fig. 3 presents the average capacity utilization achieved by different algorithms, with the approximation parameter set as 50%, 60%, 70%, 80%, and 90%, respectively. Compared with *MCF*, *Calantha2* and *Calantha3* increase the average capacity utilization by up to 17.8% and 12.3%, respectively. Our simulation results also show that *Calantha* outperforms *Blossom*, with the exception that *Calantha3* has lower average capacity utilization than *Blossom* when $\alpha = 90\%$. In case of $\alpha = 80\%$, even *Calantha* greatly reduce the number of spanning trees used in the multicast sessions, the average capacity utilization doesn't decrease.

To verity that the resulted rate allocation of our algorithm



(a) $\alpha = 60\%$ (b) $\alpha = 70\%$ (c) $\alpha = 80\%$

Fig. 2. Comparison between the number of spanning trees used in each session by *MCF*, *Blossom* and *Calantha*.



(a) $\alpha = 60\%$ (b) $\alpha = 70\%$ (c) $\alpha = 80\%$

Fig. 4. Comparison between the demand satisfaction ratio by *MCF*, *Blossom* and *Calanth*.

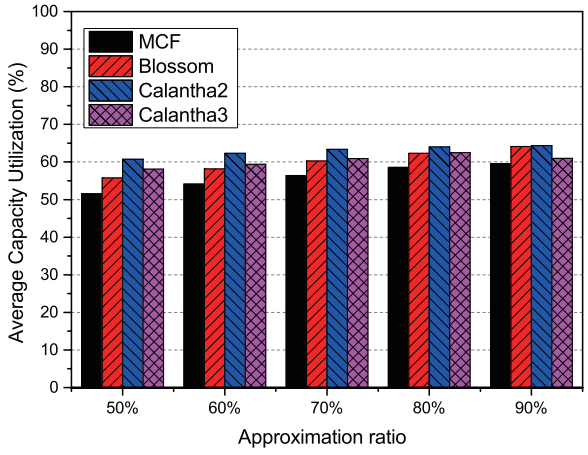


Fig. 3. Average capacity utilization in the inter-datacenter network.

spanning tree to be 2. It is probably because of the short hops make it less possibly to have link share with other sessions.

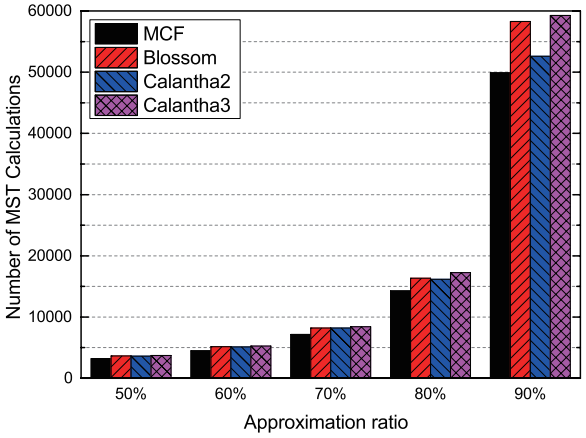


Fig. 5. Comparison of runtime (minimum spanning tree calculations).

remains max-min fair, we compared the demand satisfaction ratio achieved by each session in Fig. 4. With *MCF*, all sessions have nearly the same demand satisfaction ratio, around 1.2, as the absolute fairness must be maintained in *MCF*. In contrast, both *Calantha* and *Blossom* continually increases the rate of session 1 after other sessions are saturated due to link capacity constraints. That is to say, *Calantha* maintains the property of max-min fairness. Moreover, we find that the rates of session 4 and session 5 are increased obviously in *Calantha2* when we constrain the number of hops in a

We next compare the running time of *MCF*, *Blossom*, and *Calantha*. Since both *Blossom* and *Calantha* continually increase the rate to unsaturated sessions, we can see from Fig. 5 that *Blossom* and *Calantha* have much more minimum spanning tree calculations than *MCF*. However, *Calantha2* spends less running time than *Blossom*. In particular, when $\alpha = 90\%$, *Calantha2* has 9.8% less minimum spanning tree calculation than *Blossom*, and it has only 5.3% more

calculation than *MCF*.

In general, *Calantha* significantly reduces the number of spanning trees that are used by each multicast sessions, without sacrificing the average capacity utilization and maintaining max-min fairness among competing sessions. When $\alpha = 80\%$, *Calantha2* reduces the number of spanning trees by 44.5% as compared to *Blossom*. Besides, it has 2.7% more average capacity utilization and 1.0% less minimum spanning tree calculations.

VI. Related Work

Improving capacity utilization of inter-datacenter networks via centralized traffic engineering is an active research topic recently [2], [1]. The basic idea of these approaches is a max-min fair multicommodity flow solution. However, when each commodity becomes a multicast session that consists of a source datacenter and several destination datacenters, the problem becomes much more challenging even in a centralized way.

Airlift [3] binds the multicast session originated from the same datacenter to the same set of destination datacenters as an aggregated session. It achieves the maximal throughput 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 [21], [22], [23]. In peer-to-peer networks, distributed algorithms are desired because individual peer doesn't have a global view on network topology.

VII. Conclusion

In this paper, we focus on how content may be replicated across geo-distributed datacenters reliably. The motivation of our scheme is two-fold. First, it is not necessary to use all spanning trees to boost the rates of multicast sessions, and secondly, it is not reliable to use spanning trees with more than 3 hops. We formulate the problem as a variant of multiple concurrent commodity flow problem, by considering both the link capacity and spanning tree hop constraints. An approximation algorithm named *Calantha* is developed, which constrains the largest hops in minimum spanning trees. We use extensive trace-driven simulation to evaluate the performance of *Calantha*, and show that the algorithm can significantly reduce the number of spanning trees used for content distribution, and maintain max-min fairness among competing sessions without sacrificing capacity utilization. As part of our future work, we will implement a prototype to verify whether it is beneficial and more reliable to use hop-constrained spanning trees in a real inter-datacenter overlay network.

Acknowledgment

The authors would like to thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

References

[1] 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.

[2] 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.

[3] 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.

[4] Y. Li, H. Xie, and Y. Liao, "Blossom: Content distribution using inter-datacenter networks," in *Proc. IEEE GLOBECOM*, 2016.

[5] J. Baker, C. Bond, J. C. Corbett, J. Furman, A. Khorlin, J. Larson, J.-M. Leon, Y. Li, A. Lloyd, and V. Yushprakh, "Megastore: Providing scalable, highly available storage for interactive services," 2011.

[6] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? designing and implementing a deployable multipath tcp," in *Proc. USENIX NSDI*. USENIX, 2012.

[7] A. Balakrishnan and K. Altinkemer, "Using a hop-constrained model to generate alternative communication network design," *ORSA Journal on Computing*, vol. 4, no. 2, pp. 192–205, 1992.

[8] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *Journal of the ACM (JACM)*, vol. 37, no. 2, pp. 318–334, 1990.

[9] P. Marbach, "Priority service and max-min fairness," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 5, pp. 733–746, 2003.

[10] G. Dahl, "The 2-hop spanning tree problem," *Operations Research Letters*, vol. 23, no. 1, pp. 21–26, 1998.

[11] L. Alfandari and V. T. Paschos, "Approximating minimum spanning tree of depth 2," *International Transactions in Operational Research*, vol. 6, no. 6, pp. 607–622, 1999.

[12] L. Gouveia, "Multicommodity flow models for spanning trees with hop constraints," *European Journal of Operational Research*, vol. 95, no. 1, pp. 178–190, 1996.

[13] L. Gouveia and C. Requejo, "A new lagrangean relaxation approach for the hop-constrained minimum spanning tree problem," *European Journal of Operational Research*, vol. 132, no. 3, pp. 539–552, 2001.

[14] "Data center locations," http://bit.ly/2ntZ9Yy.

[15] G. Gamrath, T. Fischer, T. Gally, A. M. Gleixner, G. Hendel, T. Koch, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch *et al.*, "The scip optimization suite 3.2," *ZIB Report*, pp. 15–60, 2016.

[16] Y.-J. Chu and T.-H. Liu, "On shortest arborescence of a directed graph," *Scientia Sinica*, vol. 14, no. 10, p. 1396, 1965.

[17] E. Althaus, S. Funke, S. Har-Peled, J. Könemann, E. A. Ramos, and M. Skutella, "Approximating k-hop minimum-spanning trees," *Operations Research Letters*, vol. 33, no. 2, pp. 115–120, 2005.

[18] Y. Li, "Content distribution across geo-distributed datacenters," http://bit.ly/2mBLEs7, Tech. Rep., 2017.

[19] 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.

[20] "Iperf - the tcp/udp bandwidth measurement tool," http://iperf.fr/.

[21] 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.

[22] 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.

[23] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *Proc. IEEE ICNP*, 2006, pp. 2–11.