Manipulating Traffic Management in Scalable Cloud Computing Data Centers

S.Nandini,
CSE,SBM CET,
nandinisubramani12@gmail.com
Dindigul

M.D.Indu,
CSE,SBM CET,
keerthindu21@gmail.com
Dindigul

M.Kavitha
CSE,SBM CET,
kavithamuthuraman8686@gmail.com
Dindigul

***Abstract*—Cloud Computing is becoming a mainstream paradigm, as organizations, large and small, begin to harness its benefits. This novel technology brings new challenges, mostly in the protocols that govern its underlying infrastructure. Traffic en-gineering in cloud data centers is one of these challenges that has attracted attention from the research community, particularly since the legacy protocols employed in data centers offer limited and unscalable traffic management. Many advocated for the use of VLANs as a way to provide scalable traffic management, how-ever, finding the optimal traffic split between VLANs is the well known NP-Complete VLAN assignment problem. The size of the search space of the VLAN assignment problem is huge, even for small size networks. This paper introduce a novel decomposition approach to solve the VLAN mapping problem in cloud data centers through column generation. Column generation is an effective technique that is proven to reach optimality by exploring only a small subset of the search space. We introduce both an exact and a semi-heuristic decomposition with the objective to achieve load balancing by minimizing the maximum link load in the network. Our numerical results have shown that our approach explores less than 1% of the available search space, with an optimality gap of at most 4%. We have also compared and assessed the performance of our decomposition model and state of the art protocols in traffic engineering. This comparative analysis proves that our model attains encouraging gain over its peers.**

***Index Terms*—Traffic engineering, data centers, load balanc-ing, optimization, column generation.**

I. Introduction

CLOUD computing services are currently being deployed across a range of market sectors to help improve the scal-ability and cost effectiveness of Information Technology (IT) services. Data centers and networking infrastructure are key enablers for the rapid rise and adoption of cloud computing as well as a wide range of other networking applications (e.g., video-streaming, data storage, web search, etc.). Typically, data centers rely on Layer 2 switched networks [1], [2] to inter-connect tens or thousands of physical servers hosting virtual machines (VMs) belonging to various cloud providers to offer their services. Such server virtualization provided by data centers addresses some limitations of traditional data cen-ters; namely, virtualization results in higher server utilization, reduced operational cost, better application isolation and hence better performance [3].

Today, data centers are witnessing an explosive growth in the volume of data exchanged, stored and processed, and the trend of deploying applications and services on the cloud is driving the need for mega data centers. Up until 2006, Google acquired more than 450,000 servers [4], Amazon's EC2 cloud is powered by almost 500,000 servers [5]. Microsoft and Yahoo have hundreds of thousands of servers in their data centers [6]. While many praised Ethernet for being the ideal technology for intra-data center networks [2], [6], [7], [8], and [9], this technology is difficult to scale as data centers grow in size and load. Indeed, Ethernet is attractive for being ubiqui-tous, inexpensive, and off-the shelf commodity. Its ease of use, self-learning, and independent forwarding capability makes it ideal for data center network environment. The conventional traffic splitting technique in Ethernet switches is the Spanning Tree Protocol (STP). STP consists of finding a loop-free path that spans all nodes in the network. Links that violate the loop-free constraint are excluded by blocking the corresponding switch ports. To achieve all these features, Ethernet relies on broadcast-based communication and packet flooding to learn host location, which makes it highly unscalable. Moreover, the STP protocol suffers from many shortcomings [8], such as poor resource utilization, failure to exploit the path redundancy in the physical topology, lack of reliability in case of switch or link failures, etc.

Scalable traffic management in data center networks has re-cently become a factor of utmost importance [2], [8], [9], [10], [11], [12], [13], [14]. By making use of Virtual Local Area Networks (VLANs), data center networks are able to logically isolate the traffic and resources of the various applications they host. VLANs partition nodes in the network into com-munities of interest. Servers within one community can only communicate with servers that belong to the same community (or VLAN). Such practice allows both performance isolation and network scalability, since packets exchanged within a VLAN do not stretch to the entire network. Additionally, to exploit path redundancy in the network, the Multiple Spanning Tree Protocol (MSTP), an extension of STP, allows different VLANs to use different spanning trees traversing diverse physical links and switches. A traffic engineering framework for MSTP in large data centers is presented in [14] where the authors focused on the mapping of each VLAN into a

Manuscript received September 29, 2013; revised December 12, 2013. The editor coordinating the review of this paper and approving it for publication was L. Badia.

C. Assi, S. Ayoubi, and S. Sebbah are with Concordia University, Montréal, QC, H3G 2W1, Canada (e-mail: assi@encs.concordia.ca).

K. Shaban is with Qatar University (e-mail: khaled.shaban@qu.edu.qa).

This work was made possible by the NPRP 5-137-2-045 grant from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

Digital Object Identifier 10.1109/TCOMM.2014.012614.130747

spanning tree to improve the overall network utilization. Smart Path Assignment In Networks (SPAIN) is presented in [8] to provide multipath forwarding by exploiting commodity off the shelf Ethernet switches; SPAIN exploits the redundancy in an arbitrary network topology and pre-computes paths that are merged into trees which later are mapped into VLANs to achieve higher bisection throughput. Multipath routing in data center networks is also addressed through ensemble routing [11]; the framework aggregates individual flows into routing classes which are then mapped into VLANs to load balance the traffic across the network links. Traffic engineering in data centers with ensemble routing is studied in [10] where the authors leverage the multi-path routing to enable traffic load balancing and quality of service provisioning. The authors noted that a key challenge for intra-data center traffic engineering is the optimal flow mapping into VLANs (known as the VLAN assignment problem) to achieve load balancing.

The VLAN assignment problem is NP-complete with a large search space [9]. For instance, for a network with v VLANs and c flows, there are $|v|^c$ possible mappings [10]. Finding the best mapping is therefore a large-scale combinatorial problem. Several prior works have attempted to address this or simpler variation of this problem. For instance, SPAIN [8] proposed a greedy heuristic for packing flows or paths into a minimal set of VLANs and the subgraphs containing these VLANs are dynamically constructed. A constraint based local search based on Constraint Programming (CP) is presented in [14] for mapping flows into VLANs, assuming the set of VLANs is given. The authors of [10] used Markov Approximation techniques to solve the mapping problem and designed approximation algorithms with close to optimal performance guarantees. The authors assume the VLANs or the spanning trees are pre-constructed and their method assigns routing classes to these VLANs with the objective of minimizing link congestion.

In this paper, we consider the problem of traffic engineering in data center networks; namely, we characterize each tenant request by a set of VMs communicating with each other. We address the problem of mapping traffic flows of each tenant (flows between VMs) into VLANs; similar to [10], a separate spanning tree is constructed per each VLAN. As the number of spanning trees in a network could be very large (e.g., in a fully connected graph with n nodes, there are as many as n^{n-2} spanning trees), selecting the most promising spanning trees to map the traffic flows onto is a very challenging and complex combinatorial problem. This paper jointly addresses the problem of tree construction and flow mapping, a seemingly very large combinatorial problem. To keep track of the problem, we follow a primal-dual decomposition approach using Column Generation; here, the problem is divided into two subproblems (Master and Pricing), the former builds sub mappings (a single spanning tree with mapped flows), and the latter, selects among the sub mappings, the global mapping of the problem. Our method proves to be highly scalable in addressing the joint VLAN mapping and tree construction problem.

The rest of the paper is organized as follows: Section II presents a discussion of the state of the art work in data centers traffic engineering. Section III is dedicated for problem definition. Section IV introduces the column generation approach for solving the VLAN assignment problem. Section V illustrates our experimental results. We conclude the paper in Section VI and list our future work.

II. Related Work

Today's data center networks are expected to provide high network utilization to the large number of distinct services they support. These networks are also expected to provide guaranteed performance for the various hosted tenants sharing the same networking infrastructure.

Intelligent traffic engineering mechanisms inside a data center become therefore of utmost importance to achieve better load balancing and guarantees on the quality of service. Recently, the problem of traffic management in data center networks has been receiving numerous attention with the goal of providing predictable performance. The issue of performance predictability becomes of particular interest given the nature of data center topologies, which tend to be oversubscribed [2]. The authors of [15] noted that variability in network performance harms application performance and causes service provider revenue loss. As a result, they advocated to provide tenants with virtual networks connecting their computing (VM) instances. The authors proposed a virtual network abstractions which capture a tradeoff between application performance and provider costs. The abstraction is a mean to expose the tenant requirements to the provider as well as the virtual network interconnecting the VMs. The problem reduces to a problem of resource allocation of VMs into physical servers where the network manager needs to ensure that bandwidth demands can be satisfied while maximizing the number of concurrent tenants. Ultimately, the authors have shown that tenants can get predictable performance in a multitenant shared data center network infrastructure.

Similarly, the authors in [16] addressed the problem of performance predictability in multi-tenant data center networks with VMs and tasks from multiple tenants coexisting in the same cluster. As applications are largely uncoordinated and mutually untrusting, the potential for network performance in-terference and denial of service attacks is high. They presented Seawall, a proportional bandwidth allocation scheme that divides network capacity based on an administrator-specified policy. The method computes and enforces allocations by tunnelling traffic through congestion controlled paths. Seawall achieves efficient multiplexing of the underlying communica-tion infrastructure with performance isolation.

Hedera [17] is a dynamic flow scheduling system for data center networks based on multistage switch topologies. Hedera depends on a central scheduler that measures link utilization in the network and move flows from highly utilized links to less utilized ones. When the scheduler detects a flow with augmenting bandwidth demand that exceeds a certain threshold, it computes a non-conflicting path to route this flow in order to improve the bandwidth-bisection in the network. Hedera uses simulated annealing and the global first fit heuristics for path computation. When compared to the Equal Cost MultiPath (ECMP) protocol, Hedera was found to yield substantial throughput performance gains. MicroTE

[12] is a fine grained traffic engineering approach that consists of exploiting the short-term predictability in data center traffic. A central controller is used to collect and analyze traffic in data centers for traces of similarities. Then, traffic is segregated between predictable and unpredictable, where predictable traffic demands are routed optimally using a Linear Programming (LP) model, while the remaining traffic is routed using the weighted ECMP protocol. However, it is noted in [18] that such centralized solutions face scalability and fault tolerance challenges. Accordingly, the authors of [18] proposed a method for load-aware flow routing in data center networks which does not require centralized control. Their approach achieves a maximal multi-commodity flow while tolerating failures. The multi-commodity flow problem is decomposed into two subproblems, a slave dual which can be solved at the end hosts and a master dual solved locally at each switch. The issue of performance predictability is also addressed in [19] through data center network virtualization [1]. The authors proposed SecondNet which focuses on bandwidth allocation between tenant's virtual machines. A Virtual Data Center (VDC) manager is elaborated to handle the mapping of VMs onto physical resources and routing paths with bandwidth guarantees. SecondNet also supports elasticity following changes in tenants demand for resources.

Now to achieve high bisection bandwidth in data center networks, the authors of [8] proposed SPAIN, which provides multipath forwarding using Layer-2 Ethernet switches over arbitrary topologies. Here, SPAIN aims at finding multiple paths between every pair of nodes, and then grouping these paths into a set of trees, each tree is mapped as a separate VLAN onto the physical network. The path assignment into VLANs uses a greedy packing heuristic with the objective of using the minimum number of VLANs. SPAIN's performance was validated through both simulations and experiments and has shown to achieve superior goodput over STP. Traffic engineering in data center networks has been a topic of surging interests recently; particularly, exploiting path redundancy in the network and mapping paths into VLANs to achieve higher utilization and better load balancing is addressed in [14]. The authors leverage the MSTP protocol in large data centers and presented a heuristic, based on local search algorithm, to select good spanning trees to map the traffic demand matrices while achieving minimum overall link utilization. Similar to [14], the authors of [10], [20] have addressed the problem of traffic engineering in data center network by managing aggregate (ensemble) flows rather than individual flows. Essentially, the problem considered is that of mapping routing classes to VLANs to achieve load balancing. In [20], a VLAN placement algorithm is presented for growing spanning trees to reach all switches in the network and subsequently traffic splitting onto those VLANs is presented (through linear programming) to achieve load balancing. However, the heuristic construction of spanning trees does not provide any guarantees on the quality of the generated solution. More recently, the authors of [10] considered the problem of traffic engineering with ensemble routing and noted the combinatorial challenge of optimizing the assignment of routing classes into VLANs. They used the Markov approximation framework for approaching the mapping problem and developed approximation algorithms with close to optimal performance.

[1]For a recent comprehensive survey on data center network virtualization, readers are referred to [3].

III. The VLAN Assignment Problem

In this section, we present a formal definition of the VLAN assignment problem. We present two computational analysis to prove the NP-Complete nature of the problem under various considerations. We also show through an illustrative example the exponential search space that the VLAN assignment prob-lem poses.

A. Problem Definition

We consider a data center network whose underlying physi-cal topology is represented by a graph $G(N, L)$, N is the set of network nodes and L is the set of communication links. Each link connects a pair of network nodes (i, j) and has a capacity $R_{i,j}$. We assume a multi-tenant data center, each tenant re-quests virtual machines with computing and storage resources and specifies the complete pairwise bandwidth demand matrix between its VMs. Note that we do not address here the problem of embedding virtual machines into the data center, but rather, and similar to [10], we focus on the problem of mapping traffic flows between VMs into VLANs. We assume each flow has a bandwidth demand of δ_C and let C be the set of all flows and V be the set of VLANs. We acknowledge that the maximum number of VLANs a switched Ethernet network supports is 4096, however, the number of spanning trees in the network could substantially **exceed** that. Given the traffic matrix, finding the most useful spanning trees to these VLANs is a combinatorially complex and challenging problem. Thus, the VLAN assignment problem can be formulated as follows:

Problem Definition 1. *Given a network $G(N, L)$, the set of VLANs V, and a set of flows C, each with bandwidth requirement of δ_C, find the optimal mapping of flows (paths) to VLANs, such that the bisection bandwidth or goodput in the network is maximized or the maximal link utilization is minimized.*

Attempting to maximize the throughput or bisection bandwidth means admitting as many flows as possible into the network. This indeed yields better revenues for network providers by allowing them to host more cloud services. Alternatively, if one were to load balance the traffic across the network and minimize congestion to achieve better quality of service for the communicating VMs, then minimizing the maximal link load becomes the objective. This objective is necessary for building service level agreement (SLA) and QoS guarantee in data center networks. Indeed, solving the VLAN assignment problem under either of these objectives is an NP-Complete.

Theorem 1. *The VLAN assignment problem with the objective of maximizing the network capacity is NP-Complete.*

Proof: The VLAN assignment problem is polynomially reducible to the "Multiple Knapsack Problem" (MKP) [21]. The Multiple Knapsack decision problem is a variation of the 0-1 Knapsack problem, which is NP-Complete. In the MKP,

we have multiple knapsacks and we aim to place as many items as possible in each knapsack, such that the capacity of each is not violated. The MKP problem is defined as follows:

Problem Definition 2. *Given a set of m knapsacks, each with capacity γ_m, and a set of n items, each with profit π_n. Find k subset of items to be placed in each knapsack, such that total amount of profit gained is maximized, without violating the capacity of each knapsack.*

We relax the VLAN assignment problem such that all links have the same capacity, and each flow will be routed on every link in the VLAN. Note that while multiple flows can be mapped to the same VLAN, the same flow cannot be mapped to more than one VLAN. Thus, in this relaxed version of the problem, the knapsacks are the VLANs and the items are the flows. Each VLAN has a capacity equal to the uniform capacity of the links and the flows each has a bandwidth requirement. Placing a flow on a VLAN will place a bandwidth load on every link in the VLAN, and thus decreases its overall capacity. Our goal is to admit as many items (flows) as possible. Since the relaxed version of the problem is NP-Complete, adding more constraints to the problem, by having specific origin and destination for each connection and that one link can belong to multiple VLANs, is therefore NP-Complete too. ■

Theorem 2. *The VLAN assignment problem with the objective of minimizing the maximum link utilization is NP-Complete [10].*

The authors of [10] have shown that this mapping is polynomial-time reducible into the minimum-weight Steiner tree problem, which is known as NP-complete and APX-hard. In addition, the VLAN assignment problem also suffers from an exponential search space, both in the number of VLANs present in a given network, as well as in the number of different possibilities in which connection demands can be mapped to the VLANs. These two elements render the problem heavy, for the following reasons.

- First, enumerating all spanning trees in a given network can be problematic and tedious, particularly as there are no efficient algorithms to perform this enumeration, not to mention the large number of spanning trees present in a medium-sized network. For instance, for a clique topology, there are $n^{(n-2)}$ spanning trees, by the well-known Cayley's formula [9].
- Second, since the Ethernet standards can only support 4096 VLANs, a large number of the enumerated trees will never be used, and the challenge becomes to find a subset that can lead to optimal traffic split.
- Third, the number of different possibilities in which flows (or paths) can be mapped to available VLANs is exponential. This is similar to the various ways n distinct objects can be distributed into m different bins with k_1 objects in the first bin, k_2 in the second, etc. and $k_1 + k_2 + \ldots k_m = n$. This indeed is obtained by applying the multinomial theorem where $\binom{n}{k_1+k_2+\ldots+k_m=n\ k_1+k_2+\ldots+k_m} = m^n$. Therefore, for C flows and V VLANs, there are V^C different mapping possibilities.

To further convey the complexity of the problem, we consider 3 traffic flows in a complete graph K_4; in such a 4-nodes topology, there are 16 different spanning trees. Mapping the traffic flows to these spanning trees involves a large number of possibilities as shown in Figure 1. For instance, one possible solution would be to assign all demands to a single VLAN. Another feasible solution would be to map each flow to a different VLAN. Or, map two flows to the same VLAN, and place the remaining one in a different VLAN. In fact, with 16 different spanning trees to choose from, there are 16^3 (4096) different mapping possibilities. This number gets more dramatic as the network grows. For instance, in a K_5 complete graph with 3 connections, the number of mapping possibilities is approximately 2 millions!

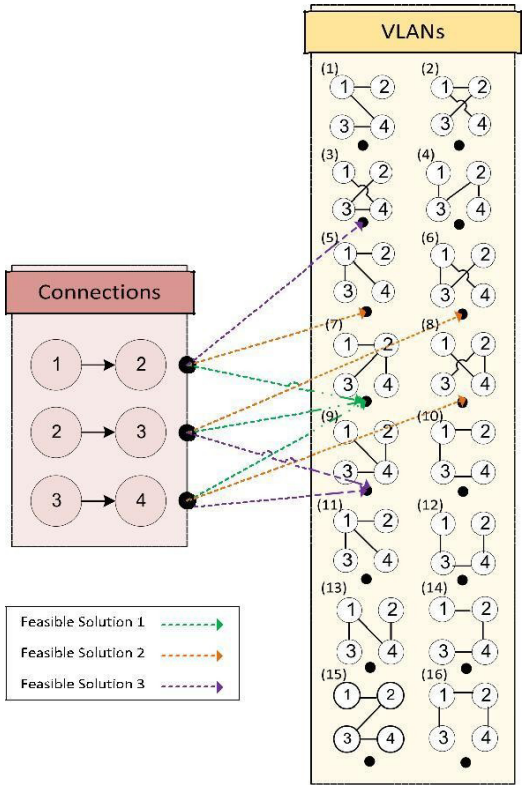


Fig. 1. Connection demands to VLAN mapping.

Clearly, while exhaustive enumeration of the spanning trees as well as the mappings will yield to the optimal solution, such enumeration is prohibitively computationally expensive. We explore in a subsequent section the possibility of only generating a small subset of such configurations or mappings. Our approach uses a column generation framework where only configurations/mappings that are deemed good are constructed through a reduced cost function (more details are presented in the sequel).

B. Problem Formulation

In this formulation, we assume the set of VLANs is predetermined (through offline enumeration). Given a traffic demand matrix between VMs, we aim at balancing the traffic load across the network through minimizing the maximum link utilization.

Below we present the mathematical model of the VLAN assignment problem:

- Parameters:

$G(N, L)$: The physical network with N nodes and L links.

R_{ij}: Capacity of Link $(i,j) \in L$.

C: Set of flows, each with an origin $o(c)$, a destination $d(c)$, and a bandwidth demand of δ_c.

V: Set of VLANs.

$\sigma_{ij}^{v} = \begin{cases} 1, & \text{if VLAN } v \text{ traverses link } (i,j), \\ 0, & \text{otherwise.} \end{cases}$

- Decision Variables:

$x_c^v = \begin{cases} 1, & \text{if flow } c \text{ is mapped onto VLAN } v, \\ 0, & \text{otherwise.} \end{cases}$

$y_{ij}^{c,v} = \begin{cases} 1, & \text{if } c \text{ mapped onto } v \text{ and routed on } (i,j), \\ 0, & \text{otherwise.} \end{cases}$

t_{ij}^{v}: total traffic contributed by VLAN v on link (i,j).

U_{ij}: Utilization of link (i,j).

The VLAN assignment problem is formulated next:

$$Min\, Max\, U_{ij}$$

Subject to

$$\sum_{j:(i,j)\in L} y_{ij}^{c,v} - \sum_{j:(j,i)\in L} y_{ji}^{c,v} \leq 1 \qquad \forall c \in C, i = o(c) \tag{1}$$

$$\sum_{j:(i,j)\in L} y_{ij}^{c,v} - \sum_{j:(j,i)\in L} y_{ji}^{c,v} = 0 \qquad \forall c \in C, i \in N \setminus \{o(c), d(c)\} \tag{2}$$

$$\sum_{j:(i,j)\in L} y_{ij}^{c,v} - \sum_{j:(j,i)\in L} y_{ji}^{c,v} \geq -1 \qquad \forall c \in C, i = d(c) \tag{3}$$

$$\sum_{v\in V} x_c^v \leq 1 \qquad \forall c \in C \tag{4}$$

$$y_{ij}^{c,v} \leq x_c^v \qquad \forall v \in V, c \in C, (i,j) \in L \tag{5}$$

$$t_{ij}^{v} = \sum_{c\in C} y_{ij}^{c,v} . \delta_c \qquad \forall v \in V, (i,j) \in L \tag{6}$$

$$\sum_{v\in V} t_{ij}^{v} \leq R_{ij} \qquad \forall (i,j) \in L \tag{7}$$

$$y_{ij}^{c,v} \leq \sigma_{ij}^{v} \qquad \forall v \in V, c \in C, (i,j) \in L \tag{8}$$

$$U_{ij} = \frac{\sum_{v\in V} t_{ij}^{v}}{R_{ij}} \qquad \forall v \in V, (i,j) \in L \tag{9}$$

Constraints (1-3) are the flow conservation for routing across the network. Constraint (4) indicates that one flow can be mapped on one VLAN only. Constraint (5) ensures that every flow will only be routed through the links that form the VLAN onto which it is mapped. This constraint establishes a strict relation between where a given flow is routed and how. Constraint (6) calculates the amount of traffic contributed by each VLAN v on a link (i,j). Constraint (7) ensures that the amount of traffic routed on a given link, does not exceed the capacity of the link. Constraint (8) ensures that a connection cannot use a link in a VLAN, if that link does not belong to the spanning tree of that VLAN. Finally, constraint (9) calculates the utilization of every link in the network.

IV. Decomposition Model

In the following section, we present our column generation model, as well as, the intuition by which we decomposed the original VLAN assignment problem into two subproblems towards a scalable solution to the aforementioned hurdle.

A. Column Generation

Column Generation is an efficient method for solving large LP problems [22]. Normally, given an LP, the algorithm begins with an initial subset of configurations (columns) (M_0) that satisfies all the constraints. At each iteration, a new configuration ($m \in M$, where M is the set of all possible configurations or columns), that ameliorates the objective function, is added to the initial set ($M_0 = M_0 \cup m$). Thus, unlike the SIMPLEX method, column generation only explores a subset of the variables, instead of enumerating all of them. In every iteration of the SIMPLEX method, the goal is to find the next non-basic variable to enter the set (basis), which is the one with the minimum reduced cost coefficient. The SIMPLEX method resorts to calculating the reduced cost coefficient of all non-basic variables whereas column generation alleviates this by only finding this next-entering variable. Column generation decomposes the initial problem into two sub-problems, a master (LP model) and a pricing (ILP). The master is in charge of determining if the explored configurations satisfies all the constraints. The pricing model is in charge of finding a new configuration to be passed on to the master. If the newly found configuration was deemed feasible (and will improve the objective) by the master, it will be added as a new column, hence the name of the method. The pricing's objective function is in fact the reduced cost coefficient of the master. Thus, the master model can be referred to as the primal, while the pricing as the dual. The master and the pricing problems alternates until no new configurations are found with a negative reduced cost coefficient (in the case of a minimization problem) which indicates that optimality is reached.

B. The Intuition Behind our Decomposition Approach

Decomposing the VLAN assignment problem to be solved using the column generation method is not straight forward. To achieve a concise and vigorous decomposition, we attempted to reformulate the problem in a way that resembles the famous cutting-stock problem [22]. In the cutting stock problem, the difficulty resides in the very large number of cuts available to choose from. This indeed resembles the VLAN assignment problem where a very large number of mapping possibilities between flows and VLANs exists.

Similar to the cutting stock problem, we envision a cut in the VLAN assignment problem as a unique mapping of a subset of flows to a particular VLAN. Finding the optimal set of cuts is nothing but finding the optimal way our various flows can be

mapped to a subset of the available VLANs. Following these pinpoints, we can divide our problem into two subproblems: the pricing will be in charge of finding a new configuration, while the master will ensure that these configurations do not violate the capacity constraints. In the following subsection, we will present our decomposition model.

C. The Column Generation Model for the VLAN Assignment Problem

We will now introduce our column generation model for the VLAN assignment problem. We have modified the problem definition to include not only finding the optimal mapping of flows to VLANs, but also finding the optimal set of VLANs (spanning trees) onto which these flows will be mapped. In fact, as the number of possible spanning trees in a network can be very large, limiting the search space to those VLANs that offer a good quality solution will greatly improve the runtime of the model. We introduce a new variable m that denotes a configuration. A configuration is defined as a VLAN v onto which flows are mapped.

Thus, the problem definition of the column generation model can now be stated as follows:

Problem Definition 3. *Given a network $G(N, L)$, and a set of flows C, each with bandwidth demand δ_c, find the optimal set of configurations, such that the maximum link utilization is minimized.*

At every iteration, the pricing model will generate a new configuration that enhances the objective function. These generated configurations will be sent to the master to ensure that they do not violate the link capacity constraints, as well as the constraint of mapping a connection onto only one VLAN. To generate a new configuration, the pricing requires building a spanning tree and mapping flows onto the generated trees. Each new configuration has to improve the master's objective function and the reduced cost coefficient. The pricing will keep running until its objective function becomes greater than or equal to 0 (in the case of a minimization problem) which indicates that optimality is reached. The master problem can be formulated as follows:

The Master Problem:

- Parameters:

$G(N, L)$: the network with N nodes and L links. C: set of flows, each with an bandwidth demand δ_c. R_l: Capacity of link $l \in L$.
M: set of *all* configurations, each indexed by m. M_0: set of explored configurations.

$a^c_m = \begin{cases} 1, & \text{if flow } c \text{ is mapped to configuration } m, \\ 0, & \text{otherwise.} \end{cases}$

b^l_m represents the traffic flow contributed by configuration m on link l.

- Decision Variables:

$z_m = \begin{cases} 1, & \text{if configuration } m \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases}$

α: represents the maximum link utilization.

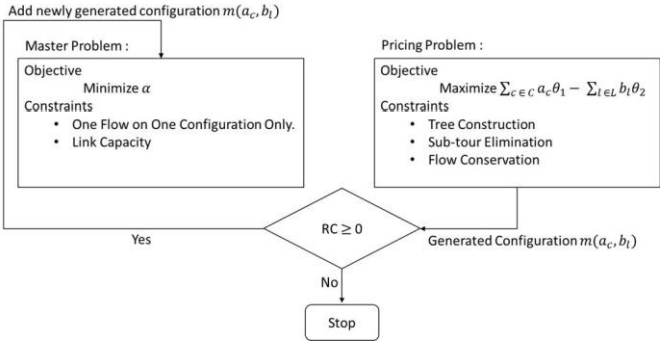


Fig. 2. Flow chart of CG1 model.

$$Minimize\ \alpha$$

Subject to

$$\sum_{m \in M_0} a^c_m z_m = 1 \qquad \forall c \in C \tag{10}$$

$$\sum_{m \in M_0} b^l_m z_m \leq \alpha \qquad \forall l \in L \tag{11}$$

$$z_m \in \{0,1\} \qquad \forall m \in M_0 \tag{12}$$

$$\alpha \leq 1 \tag{13}$$

The objective of the master problem is to achieve load balancing by minimizing the maximum link utilization. However, we reformulate the min-max constraint by defining a new variable α that represents the worst link utilization, and we aim to find the optimal value α such that none of the links in the network have a utilization that exceeds this value. Constraint (10) ensures that each flow, if mapped, can only be running on one configuration or one VLAN. Constraint (11) ensures that the physical capacity of the links in the network is not violated. Constraint (12) is the integrality condition that indicates if a configuration m is selected or not. However, for solving the master, we relax this constraint and allow z_m to take any value in the range between 0 and 1. This becomes an LP relaxation of the original problem and when solved yields a lower bound solution to the original problem. Finally, constraint (13) ensures that α could be at most equal 1, which indicates that at least one link in the network is fully saturated.

The Pricing Problem:

Note that a^c_m and b^l_m are both parameters in the Master problem which are obtained after solving the pricing subproblem. During every iteration, when the master problem is solved, we need to verify the optimality of the solution. If it is optimal we conclude our search, or else decide a new column to join in its current basis which may improve the current solution. This can be achieved by examining whether any new configuration that has not been added to the basis has a negative reduced cost. Denote the dual variables corresponding to (10) and (11) by Θ_1 and Θ_2 respectively. The reduced cost (RC) for an off basis column is expressed as:

$$RC = \sum_{c \in C} ac\,\Theta\,1 - \sum_{l \in L} bl\,\Theta 2 \quad (14)$$

When the Master's objective is to minimize, the standard pivoting rule of the simplex method is to choose a new column (configuration) such that C ac Θ1 − L bl Θ2 isc=1l=1 maximum; the column (configuration) that is found is added to the basis of the master model. The master model is solved, again, with the new basis to obtain a new solution, and the dual variable is passed to the pricing which is again solved. The two subproblems are solved iteratively until there is no off-basis column with a positive reduced cost found and therefore the solution is optimal. Indeed, this requires that the last simplex iteration of the pricing model is solved to optimality to ensure that there is no off basis column with positive reduced cost remains unexplored. Figure 2 presents an illustration of how the master and the pricing subproblems work iteratively and jointly until the optimal solution is found ($RC \leq 0$).

Before we present the pricing model, we introduce a new set of decision variables for the pricing problem that are needed for spanning tree construction. These decision variables are listed below:

• Decision Variables:

$$ri = \begin{cases} 1, \text{ if node } i \text{ is the root node of the tree } T, \\ 0, \text{ otherwise.} \end{cases}$$

$$y_i^j = \begin{cases} 1, \text{ if node } j \text{ is the parent node of } i \text{ in } T, \\ 0, \text{ otherwise.} \end{cases}$$

$$Vi = \{ j \in N : l = \{i, j\} \in L \}, \forall i \in N.$$

$$xc_{ij} = \begin{cases} 1, \text{ if flow } c \text{ is routed on link } (i, j), \\ 0, \text{ otherwise.} \end{cases}$$

$$ac = \begin{cases} 1, \text{ if flow } c \text{ is routed,} \\ 0, \text{ otherwise.} \end{cases}$$

bl represents the traffic flow on link l.

Thus, the mathematical model of the pricing problem can be stated as follows:

$$\text{Maximize} \quad RC$$

Subject to

$$\sum_{i \in N} ri = 1 \quad (15)$$

$$\sum_{j \in Vi} y_i^j + r_i = 1 \quad \forall i \in N \quad (16)$$

$$y_i^j + y_j \leq 1 \quad \forall (i, j) \in L \quad (17)$$

$$\sum_{i,j \in V} y_i^j + y_j \leq |V| - 1 \quad V \subseteq N, i, j \in V \quad (18)$$

$$\sum_{j:(i,j) \in L} xc_{ij} - \sum_{j:(j,i) \in L} xc_{ji} \leq 1 \quad \forall c \in C, i = o(c) \quad (19)$$

$$\sum_{j:(i,j) \in L} xc_{ij} - \sum_{j:(j,i) \in L} xc_{ji} = 0 \quad \forall c \in C, i \in N \setminus \{o(c), d(c)\} \quad (20)$$

$$\sum_{j:(i,j) \in L} xc_{ij} - \sum_{j:(j,i) \in L} xc_{ji} \geq -1 \quad \forall c \in C, i = d(c) \quad (21)$$

$$\sum_{c \in C} xc_{ij} \leq 1 \quad \forall (i, j) \in L \quad (22)$$

$$bl = \sum_{c \in C} xc\,\delta_{ij} \quad \forall (i, j) \in L, sl = i, dl = j \quad (23)$$

$$y_i^j xc \leq y_i + y_{ji} \quad \forall (i, j) \in L, c \in C \quad (24)$$

$$ac = xc_{ij} \quad \forall c \in C, i = o(c) \quad (25)$$

(15) ensures that a tree has a single root. Constraints (16-18) model the spanning tree construction. Constraint (16) ensures that each node, except the root node, will have a parent node. Constraint (17) prevents cycling parent-son relationship. Constraint (18) represents the sub-tour elimination constraint. Constraints (19-21) are the flow conservation constraints. Constraint (22) ensures that a link can only accommodate a single flow in this particular configuration. This constraint, on one hand, can limit the number of generated configurations that violates the link capacity constraint in the master, and on the other hand, allows us to further decompose the pricing problem at each iteration. Constraint (23) calculates the amount of flow routed on each link l. Constraint (24) ensures that a link cannot be used if it does not belong to the spanning tree. Finally, constraint (25) indicates whether a flow c is routed or not.

Note that, once the relaxed (fractional) VLAN mapping problem is solved, as mentioned earlier, the obtained optimal solution is a lower bound to the solution of the original problem. To obtain the ILP solution, we solve the Master program one last time with zm assuming integer values and this allows us to obtain a solution to the integral mapping problem. We acknowledge however that the obtained solution is only an approximation of the optimal one and better quality solutions may be obtained by employing branch and bound methods (which at this time is left for our future work).

D. The modified Column Generation Model

While the CG decomposition substantially overcomes the computational complexity of the original mapping problem by avoiding the complete enumeration of the flow-VLAN mappings, it is to be noted that the pricing still exhibits some scalability issues given the ILP nature of its subproblem. In this section, we try to further enhance the running time of the proposed CG. Namely, we modify our pricing model by relaxing the tree construction constraints. In fact, finding a tree requires the model to first choose a single root among N available nodes. Upon the selection of a root, each node has to select one of its eligible neighbours as a parent. With a choice of V adjacent nodes for each node out of N , this results in |N |V possible combinations. As the size of the network grows, the number of tree construction constraints increases substantially. Instead of searching for a new tree at every Master-Pricing iteration, we opt to enumerate spanning

$$RC = \sum_{c \in C} a_c \Theta_1 - \sum_{l \in L} b_l \Theta_2 \quad (14)$$

When the Master's objective is to minimize, the standard pivoting rule of the simplex method is to choose a new column (configuration) such that $\sum_{c \in C} a_c \Theta_1 - \sum_{l \in L} b_l \Theta_2$ is maximum; the column (configuration) that is found is added

variable is passed to the pricing which is again solved. The two subproblems are solved iteratively until there is no off-basis column with a positive reduced cost found and therefore the solution is optimal. Indeed, this requires that the last simplex iteration of the pricing model is solved to optimality to ensure that there is no off basis column with positive reduced cost remains unexplored. Figure 2 presents an illustration of how the master and the pricing subproblems work iteratively and jointly until the optimal solution is found ($RC \leq 0$).

Before we present the pricing model, we introduce a new set of decision variables for the pricing problem that are needed for spanning tree construction. These decision variables are listed below:

- Decision Variables:

$y_i = 1$, if node i is the root node of the tree T, $y_i = 0$, otherwise.

$y_{ji} = 1$, if node j is the parent node of i in T, $y_{ji} = 0$, otherwise.

$V_l = \{j \in N : l = \{i,j\} \in L\}$, $\forall i \in N$.

$x_{ij}^c = 1$, if flow c is routed on link (i,j), $x_{ij}^c = 0$, otherwise.

$a_c = 1$, if flow c is routed, $a_c = 0$, otherwise.

b_l represents the traffic flow on link l.

Thus, the mathematical model of the pricing problem can be stated as follows:

$$Maximize \quad RC$$

Subject to

$$\sum_{i \in N} y_i = 1 \quad (15)$$

$$\sum_{j \in V_i} y_{ji} + y_i = 1 \quad \forall i \in N \quad (16)$$

$$y_i + y_{ji} \leq 1 \quad \forall (i,j) \in L \quad (17)$$

$$\sum_{i,j \in V} y_{ji} + y_{ij} \leq |V| - 1 \quad V \subseteq N, i,j \in V \quad (18)$$

$$\sum_{j:(i,j) \in L} x_{ij}^c - \sum_{j:(j,i) \in L} x_{ji}^c \leq 1 \quad \forall c \in C, i = o(c) \quad (19)$$

$$\sum_{j:(i,j) \in L} x_{ij}^c - \sum_{j:(j,i) \in L} x_{ji}^c = 0 \quad \forall c \in C, i \in N \setminus \{o(c), d(c)\} \quad (20)$$

$$\sum_{j:(i,j) \in L} x_{ij}^c - \sum_{j:(j,i) \in L} x_{ji}^c \geq -1 \quad \forall c \in C, i = d(c) \quad (21)$$

$$x_{ij}^c \leq 1 \quad \forall (i,j) \in L \quad (22)$$

$$b_l = \sum_{c} x_{ij}^c \quad \forall (i,j) \in L, l = i,j \quad (23)$$

$$x_{ij}^c \leq y_i + y_{ji} \quad \forall (i,j) \in L, c \in C \quad (24)$$

$$a_c = x_{ij}^c \quad \forall c \in C, i = o(c) \quad (25)$$

(15) ensures that a tree has a single root. Constraints (16-18) model the spanning tree construction. Constraint (16) ensures that each node, except the root node, will have a parent node. Constraint (17) prevents cycling parent-son relationship. Constraint (18) represents the sub-tour elimination constraint. Constraints (19-21) are the flow conservation constraints. Constraint (22) ensures that a link can only accommodate a single flow in this particular configuration. This constraint, on one hand, can limit the number of generated configurations that violates the link capacity constraint in the master, and on the other hand, allows us to further decompose the pricing problem at each iteration. Constraint (23) calculates the amount of flow routed on each link l. Constraint (24) ensures that a link cannot be used if it does not belong to the spanning tree. Finally, constraint (25) indicates whether a flow c is routed or not.

Note that, once the relaxed (fractional) VLAN mapping problem is solved, as mentioned earlier, the obtained optimal solution is a lower bound to the solution of the original problem. To obtain the ILP solution, we solve the Master program one last time with z_m assuming integer values and this allows us to obtain a solution to the integral mapping problem. We acknowledge however that the obtained solution is only an approximation of the optimal one and better quality solutions may be obtained by employing branch and bound methods (which at this time is left for our future work).

D. The modified Column Generation Model

While the CG decomposition substantially overcomes the computational complexity of the original mapping problem by avoiding the complete enumeration of the flow-VLAN mappings, it is to be noted that the pricing still exhibits some scalability issues given the ILP nature of its subproblem. In this section, we try to further enhance the running time of the proposed CG. Namely, we modify our pricing model by relaxing the tree construction constraints. In fact, finding a tree requires the model to first choose a single root among N available nodes. Upon the selection of a root, each node has to select one of its eligible neighbours as a parent. With a choice of V adjacent nodes for each node out of N, this results in $|N|^{|V|}$ possible combinations. As the size of the network grows, the number of tree construction constraints increases substantially. Instead of searching for a new tree at every Master-Pricing iteration, we opt to enumerate spanning

trees offline (using Dijkstra's algorithm) and let the pricing select suitable ones which improve the value of the objective of the Master's subproblem, until no further improvements (no spanning tree will yield positive reduced cost) are obtained. In the worst case, the pricing will navigate through the whole set of pre-determined spanning trees. To guarantee a well diversified set, every time the algorithm returns a tree, we increase the weight of the links in the generated tree. This ensures that our algorithm will try to avoid these links during the next tree construction.

Algorithm 1 GenerateSpanningTrees Algorithm

```
Given:
G(N, E) : an arbitrary topology
k: number of Spanning Trees to be generated
increment: edge weights increment value

ST= { };

for (e ∈ E) do /* Initialize the weight of all edges*/
    w(e) = 1;
end for

while (|ST| ≤ k) do
    t= { };

    /* Select a random node to become the root node*/
    s= Random(N);
    N̄ = N- {s};

    for (d ∈ N̄) do
        p= Dijkstra(s, d, E);
        t= t ∪ p;
    end for

    if (ST.contains(t)) then
        Break;
    end if

    if (containsCycle(t)) then
        continue;
    else
        for (e ∈ t) do
            w(e) += increment;
        end for
        ST= ST ∪ t;
    end if

end while; return
    ST;
```

The *SpanningTreeGeneration* function is illustrated in Algorithm 1. The algorithm takes as input the number of the spanning trees that needs to be generated, denoted by k, and a constant number denoted by *increment*, that represents the value to be used to increment the link weights at the end of every tree construction iteration. This constant is set to a large value in order to ensure that a link will only be included in multiple spanning trees if there are no other detours. At the end, the algorithm returns the set all spanning trees found denoted by ST. The algorithm begins by setting the link weights of all edges in the network to 1. Next, the algorithm will loop k times until k spanning trees are constructed. If at a given iteration, a redundant spanning tree t is found (meaning it already belongs to the set ST), the algorithm terminates, as it can no longer find unique spanning trees. At the beginning of every tree construction iteration, the algorithm will pick a random source from the set of nodes in the network to become the root node (denoted by s). Next, the model will run the Dijkstra algorithm to find the shortest path from the root node s to all other nodes in the network. At every iteration, the set of shortest paths are aggregated to form a spanning tree t. This guarantees that the tree will not contain any cycles. If the constructed tree is unique, the algorithm will increment the weight of every edge in that tree, and then will add that tree to the set ST.

Algorithm 2 Modified Column Generation Approach

```
Given G(N,E) /*an arbitrary topology*/
ST = GenerateSpanningTrees;
RC = -μ/*μ is a very large positive number*/
M0 = { };
while (!Terminate) do
    st = ST.next();
    /*Initialize Set of Configurations for spanning tree st*/
    Mst = { };
    while ((RC ≤0)) do
        RC = Run Master;
        /*m is a new Configuration*/
        m = Run Pricing(RC,st);
        RC = m.RC;
        Mst = Mst ∪ m;
    end while
  if (Mst.length ≤ 1) then
        Terminate = TRUE;
    end if
    M0 =M0 ∪ Mst;
end while
```

Algorithm 2 illustrates the methodology of our heuris-tic model: Given the topology of the substrate network, represented by G(N,E), the model begins by calling the *GenerateSpanningTree* function to enumerate multiple spanning trees that will be stored in a dedicated set ST. As we have previously mentioned, the *GenerateSpanningTree* function adopts the Dijsktra algorithm with link weights increment. Next, at the beginning of every iteration of the pricing model, we provide the pricing with a new spanning tree st from the set ST, and for every tree, we initialize a set Mst that represents the list of configurations for that given tree. The master and the pricing iterate over the same given tree, as long as it produces feasible configurations (a configuration with positive reduced cost). At the end of every iteration, the explored configuration m is added to the set Mst. Once the tree becomes non-bearing (no feasible configurations can be found), a new tree from the initial set is selected.

This procedure persists until one tree returns an infeasible configuration in the first trial, meaning that its M_{st} remains empty at the end of the first Master/Pricing iteration, in that case, the program terminates. Our numerical results have shown that our heuristic model achieves a great improvement in runtime and scalability over our initial exact decomposition model without sacrificing the quality of the final solution.

V. Performance Evaluation

In this section, we numerically evaluate the proposed VLAN mapping methods. In particular, we present comparisons be-tween the pure ILP and its decomposition variations. We refer to the column generation models presented in sections IV-C and IV-D as CG1 and CG2. The objective of our comparisons is to study the effectiveness of the designed methods in terms of quality of the obtained solutions and runtime. We also present comparisons with state of the art traffic engineering methods in data centre networks; namely, we compare the performance of STP, ECMP and SPAIN with the results obtained from our proposed methods. The purpose of this study is to see how well these protocols compare to the optimal obtained results. All our numerical evaluations are conducted using CPLEX version 12.4 on a pentium IV machine at 3.4 GHz with 8 GB RAM.

A. Numerical Results

We start by evaluating how our model perform when compared to the pure ILP model of section III-B. We consider a clique topology as it allows to determine the number of all possible spanning trees and configurations, which would provide a solid benchmark for comparison. We suppose each node in the clique represents a switch, and that each switch is connected to one host with an immediate link. We consider a traffic demand matrix of 500 sessions/flows, each pair of communicating nodes is randomly chosen and each demand is of one unit (normalized demand). As our focus is to map flows onto VLANs while balancing the traffic across the network (traffic engineering problem), we assume links with enough capacity to route all the demands.

Table I summarizes our findings. We have presented the average of 10 executions for each topology. The results show that, the number of configurations explored by our decomposition models are substantially smaller when compared to the search space of the pure ILP model. In fact, for a K_3, the pure ILP model navigates through 3^{500} configurations, while CG1 and CG2 explore 452 and 548 configurations respectively (observe that the fraction of explored configurations by our decomposition models is less than 1% of the search space of the pure ILP model and this fraction becomes negligible as we experiment with larger clique topologies). Next, we look at the run time of the models. We notice that as the size of the network increases, the run time of the pure ILP increases exponentially, while those of CG1 and CG2 increase somehow linearly. This is attributed to the fact that both CG1 and CG2 examine much smaller number of configurations (only good configurations are enumerated by the pricing subproblem) for solving the mapping problem. It is interesting to note that for smaller size networks, the pure ILP exhibits faster run times.

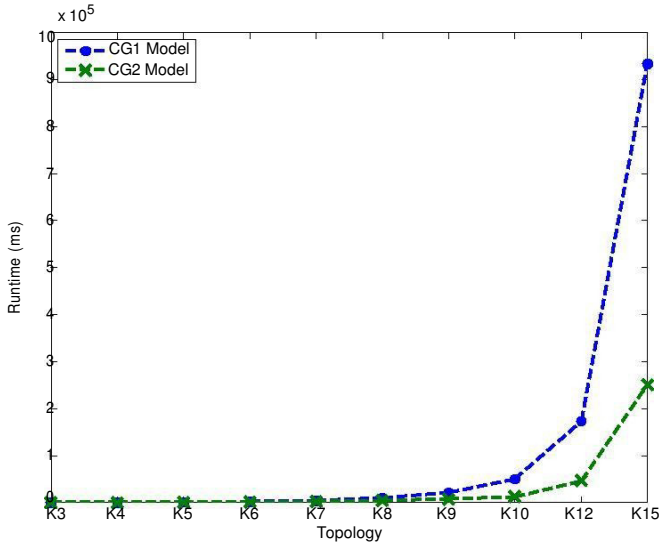


Fig. 3. Runtime results: comparison between CG1 model and CG2 model.

Indeed, this is due to the fact that with the pure ILP, the trees are always enumerated offline and the mathematical model solves the mapping problem. However, both with CG1 and CG2, the trees are explored dynamically as we seek to improve the quality of the solution. Notice that for a K_6 network, the pure ILP failed to obtain a solution after running the model for 22 hours. We also observe that CG2 consistently outperforms CG1 in terms of running time; this is due to the fact that the pricing model of CG2 is much smaller than that of CG1. Indeed, as mentioned in section IV-D, the tree construction in CG2 is not done within the pricing model (as in CG1), but rather, trees are enumerated offline and explored dynamically as we search for best configurations. The size of the ILP pricing of CG1 is much bigger and thus less scalable in terms of running time. Figure 3 compares the run times of both CG1 and CG2 over larger clique topologies of 3 to 15 nodes with all to all connection demands. The figure shows that the runtime of CG2 is much faster than CG1, in fact as we move towards cliques with more than 8 nodes, CG2 becomes at least three times faster.

Finally, we examine the quality of the obtained solutions. We show, in Table I, the % gap of the solution obtained by CG2 and CG1 to the one obtained by the pure ILP. In most cases, the gap is less than 2% for CG1 and 4% for CG2. The gap between the solution of CG1 and that of the pure ILP is attributed to the manner through which we obtained the ILP solution for CG1. Namely, as we have solved the fractional mapping problem (lower bound), we resorted to a straight forward approach for solving its ILP version. We believe this gap may be reduced if we employ a more effective branch and bound method for obtaining the integral solution. Now the gap of CG2 is attributed to the heuristic nature of the methodology followed for solving the mapping problem. That is, the more spanning trees are explored, the better the quality of the solution gets, but that comes at the cost of increased run time of the model.

TABLE I
EXPLORED SEARCH SPACE, RUNTIME AND QUALITY OF OBTAINED SOLUTION - COMPARISON BETWEEN ILP, CG1 AND CG2 (500 FLOWS)

Topology	ILP Model		CG 1			CG 2		
	#Config	Runtime(s)	#Config	Runtime(s)	Gap	#Config	Runtime(s)	Gap
K_3	3^{500}	5	452	230	1%	548	120	0%
K_4	16^{500}	24	579	318	2%	760	266	2%
K_5	125^{500}	452	660	654	2%	866	455	3%
K_6	1296^{500}	81823 >	1111	1020	2%	1322	546	4%

B. Comparative Analysis

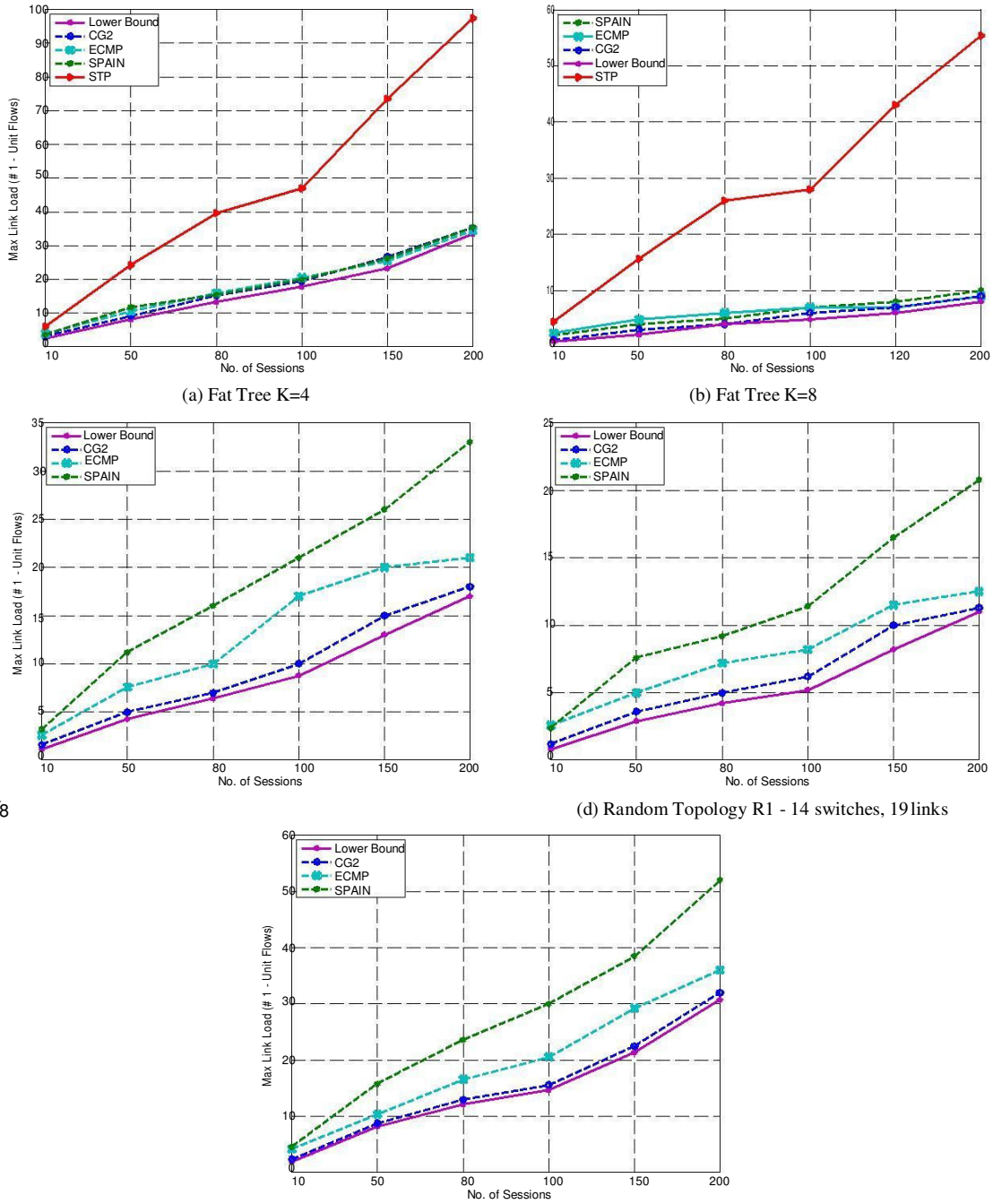
In this section, we study the performance of recent traffic engineering protocols for data center networks (SPAIN and ECMP) as well as the spanning tree protocol (STP) against the obtained solutions from our design methodology. We used CG2 (owed to its better scalability) for obtaining benchmark solutions. We use STP as a base of comparison, since it is widely used in Ethernet networks. Our objective is to study how well the traffic is balanced in the network and to measure the overall network goodput. For each experiment, we executed multiple test cases for different number of sessions in the range [10,50,80,100,150,200]. In each session, we consider random traffic demands between random pair of hosts. In all test cases, the results are averaged over multiple runs.

1) Load Balancing: We begin by studying our model's load balancing capability. For this purpose, we use the Fat Tree network, since it is a well-adopted topology for cloud data centers [2]. A Fat Tree network consists of 3 layers of interconnected ethernet switches. The links connecting the switches are of 10 GigE, while those connecting the server racks to the second layer aggregate switches are of 1 GigE, hence the name of this topology. For our experiment, we have alleviated the links that connects the second layer aggregate switches to the server racks, and considered only the three layers of interconnected switches as our substrate network.

We compared the performance of the legacy STP protocol, ECMP and SPAIN, against results obtained from the CG2 model using a Fat Tree network (K=4 and K=8) and the results are illustrated in figure 4(a) and 4(b) respectively. It is clear that the STP protocol has inferior performance, achieving very high link load. This is indeed expected, since STP concentrates the load on a particular set of links in the network and does not benefit from the existence of multipaths, thus quickly congesting the selected links. On the other hand, both ECMP and SPAIN spread equally the load across the network, achieving very close performance to those results obtained by CG2.

Next, we study further the quality of the solutions obtained by ECMP and SPAIN and compare their solutions to those obtained by CG2-LP (lower bound) and CG2-ILP (CG2). The results are shown in Figures 4(a)-4(b). We observe that for a Fat tree network (K=4, K=8), both ECMP and SPAIN performs quite well; namely, we see that the CG2 solution has a gap of 11% to the lower bound while ECMP and SPAIN both have a gap of 16%. For a Fat Tree with K=8 (Figure 4(b)), the CG2 model provides an overall 16% gap from lower bound, while ECMP and SPAIN both show a 30% gap. It is to be noted that as the number of sessions increases, the gap decreases. For instance, for Fat Tree with K=4 and 200 sessions (Figure 4(a)), ECMP's gap to the lower bound reaches 2%, while CG2 and SPAIN's gap are at 5%. This indeed shows that ECMP and SPAIN yield outstanding performance in a Fat Tree network and this is due to their capabilities in exploiting the existence of multi (redundant) paths in the network. It is important to note however that, although ECMP achieves good load balancing in the Fat Tree topology, it has been found to be unsuitable for ethernet data center networks, particularly since ECMP requires IP addressing to determine the location of the destination host. This is indeed problematic in the case where the physical topology differs from the logical topology, such as the case of VLANs. Moreover, ECMP incurs latencies due to the need to determine the next-best-hop at every switch along the path to the destination host, as opposed to pre-configured paths, where the optimal trajectory from source to destination is predetermined. In addition, its memory requirement is high, since every switch needs to keep multiple paths for every pair of nodes, which can lead to oversized routing tables. Finally, ECMP was also found to be congestion-prone, since it relies on hashing functions to determine the path to the destination host. Hashing collisions, [17] particularly between elephant flows, causes hot spots that can ultimately lead to failure in the network. Moreover, ECMP is only capable of exploring equal cost paths. Such a limitation inhibits ECMP from achieving optimal load balancing in topologies where the nodes are interconnected by multiple distinct paths of unequal cost. The interconnection of switches in the Fat Tree topology does not portray this limitation, which makes ECMP very well suitable for such a topology.

To confirm our observation, we study how well both ECMP and SPAIN perform in other network topologies. Namely, we consider a clique and two random topologies that we have generated using Brite [23]. The results are shown in Figures 4(c)-4(e). The figures show results obtained from ECMP, SPAIN, and CG2 as well as the LP relaxation of the CG mapping problem. We observe that, unlike in the fat tree network, the performance of both ECMP and SPAIN is far from optimal. For example, while the gap between CG2 and the lower bound optimal solution is 13% (Figure 4(c)), the gap between ECMP stands at around 40% from the lower bound. Similar findings are shown as well for random networks (Figures 4(d) and 4(e)). The rationale behind this is the fact that the clique topology, as well as, the random networks dispose more than 6 redundant paths between every pair of nodes, while ECMP was only capable of finding 1 equal cost path between every pair of nodes. This gain in the number of explored redundant paths is reflected in the better results obtained by CG2. With



(a) Fat Tree K=4

(b) Fat Tree K=8

(c) Clique K_8

(d) Random Topology R1 - 14 switches, 19 links

(e) Random Topology R2 - 11 switches, 25 links

Fig. 4. Max link load: comparison between CG2, ECMP, SPAIN, and STP using Fat Tree, Clique K_8, and random topologies.

respect to SPAIN, we notice inferior performance (as the figures show) in terms of achieving minimum link load for the three networks we studied. The reason is that SPAIN relies on a randomized approach when mapping flows to VLANs. This randomized approach leads to poor load balancing capabilities, especially when SPAIN exploits the existence of multiple paths with varying number of hops. Hence, a path with larger number of hops is equally likely to be utilized as a path with smaller number of hops. This leads to consuming more bandwidth from the network and yield to links with larger number of flows routed through them. Our CG2 strikes a balance between SPAIN and ECMP. It exploits the existence of multiple redundant paths by only using those paths that improve the quality of the solution and achieve better load balancing. It is important to note that the authors of SPAIN mention the possibility of including a centralized control to enhance the traffic engineering capabilities of their flows to VLANs mapping algorithm.

To observe how the traffic is balanced throughout the network, we plot the Probability Density Function (PDF) of the link loads in Figure 5. First, we observe that the CG2's PDF stops at 9, since it represents the max-link load achieved by CG2. We also observe that the CG2 model achieves better fairness in link utilization, which is reflected in the narrow

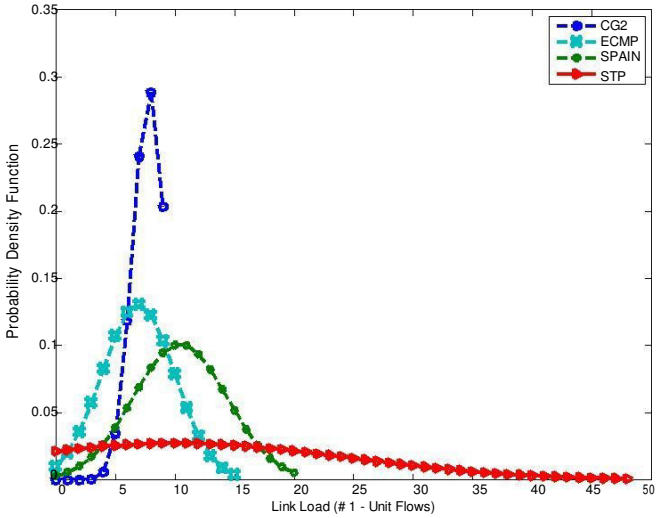


Fig. 5. Probability density function: comparison of link load between CG2, ECMP, STP and SPAIN over random topology of 14 nodes and 19 links.

bell shape of its PDF graph. However, by looking at the other protocols, we notice that their link loads fluctuate between highly utilized links and under-utilized links, indicating poor fairness. Clearly, as we have explained above, STP has the worst load distribution, with a max link load of 45, while CG2 shows the best link load distribution.

2) Goodput: In order to evaluate the network goodput, we have replaced the Master's objective function by Max $v_c \; x^{v}_{c} \delta_c$, and assigned a link capacity of 10GigE and random flow demands of 1 to 3 Gb/s. We aim to measure the amount of goodput provided by the CG2 model in comparison with ECMP, STP and SPAIN over a Fat Tree, clique and random topology. The results are shown in Figures 6(a)-6(c); the upper bound results are obtained by solving a relaxed version of the maximization problem in CG2. Clearly, for a fat tree network (Figure 6(a)), both ECMP and SPAIN achieve outstanding performance, for the same results mentioned in our previous discussions, while STP exhibits the lowest goodput. We also observe that for a clique topology (Figure 6(b)), our CG2 model outperforms ECMP and SPAIN with a 5% gap from the upper bound solution, while ECMP and SPAIN present a 7% and 13% gap respectively. Finally, in a random topology of 14 nodes and 19 links (Figure 6(c)), our CG2 model achieves a 7% overall gap, while ECMP and SPAIN provide a 22% and 25% gap, respectively.

VI. Conclusion and Future work

In this paper, we introduced a novel column generation approach for solving the VLAN assignment problem in cloud data centers. We present two decomposition approaches : an exact, as well as a semi-heuristic model to attain better runtime and scalability. We compare both models against the pure ILP model of the VLAN assignment problem, and prove that our approach yields a substantial decrease in the size of the explored search space with encouraging optimality gap. We also compared our decomposition approach against state of the art protocols in traffic engineering, our comparative analysis has shown that our model outperforms its peers in most network topologies in terms of link load, attainable gap from lower bound LP solution, as well as in goodput. As we have previously mentioned, employing a more effective technique to go from the relaxed LP solution to the integral ILP solution can potentially improve our model's optimality gap. Hence, for our future work, we aim to employ branch-and-bound to affirm this proclamation. Also, in our current work, we have assumed that the traffic demands are given. VLAN assignment problem with unknown traffic demands is a more challenging problem, thus as future work, we aim to build on our current findings and devise an efficient online decomposition model that is suitable to the bursty, sporadic and unpredictable nature of demands in cloud data centers. For future work, we also plan to extend our mapping framework to consider delay sensitive flows, multicast flows, dynamic flows, etc.

References

[1] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Commun. Rev.*, vol. 38, pp. 63–74, 2008.

[2] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," *ACM SIGCOMM Computer Commun. Rev.*, vol. 39, pp. 51–62, 2009.

[3] M. F. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: a survey," 2012.

[4] U. Hlzle, "Google's data centers: an inside look." Available: http://googleblog.blogspot.ca/2012/10/googles-data-centers-inside-look.html, Oct. 2012.

[5] R. McMillan, "Amazon cloud powered by almost 500,000 servers." Available: http://www.wired.com/wiredenterprise/2012/03/amazon-ec2/, 2012.

[6] C. Guo, H. Wu, K. Tan, L. Shiy, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," *ACM SIGCOMM Computer Commun. Rev.*, vol. 38, pp. 75–86, 2008.

[7] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks.* Access Online via Elsevier, 2004.

[8] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "Spain: COTS data-center ethernet for multipathing over arbitrary topologies," in *Proc. 2010 USENIX Conference on Networked Systems Design and Implementation*, pp. 265–280.

[9] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM Computer Commun. Rev.*, vol. 39, pp. 39–50, 2009.

[10] Z. Shao, X. Jin, W. Jiang, M. Chen, and M. Chiang, "Intra-data-center traffic engineering with ensemble routing," in *Proc. 2013 IEEE Infocom.*

[11] M. Schlansker, Y. Turner, J. Tourrilhes, and A. Karp, "Ensemble routing for datacenter networks," in *Proc. 2010 ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, p. 23.

[12] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: fine grained traffic engineering for data centers," in *Proc. 2011 COnference on Emerging Networking EXperiments and Technologies*, p. 8.

[13] C. Kim, M. Caesar, and J. Rexford, "Floodless in SEATTLE: a scalable Ethernet architecture for large enterprises," *ACM SIGCOMM Computer Commun. Rev.*, vol. 38, pp. 3–14, 2008.

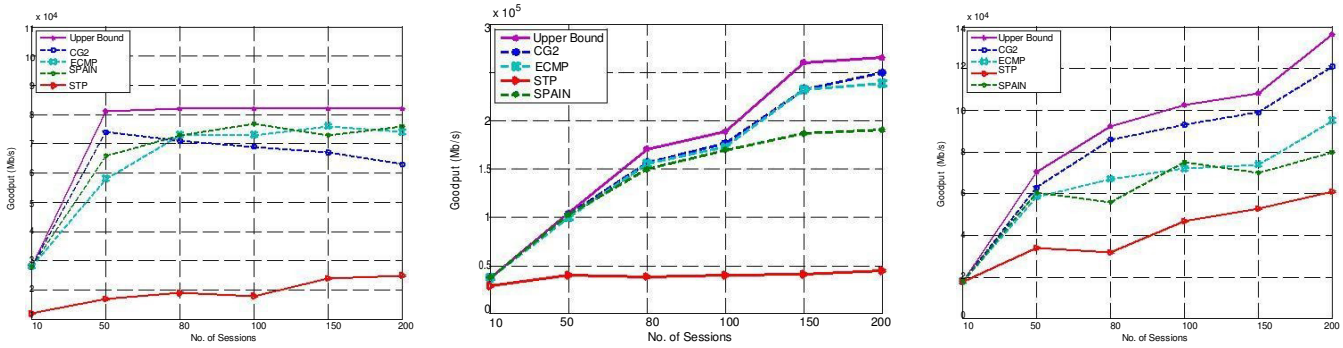
[14] H. T. Viet, Y. Deville, O. Bonaventure, and P. Francois, "Traffic engineering for multiple spanning tree protocol in large data centers," in *Proc. 2011 IEEE Internationsl Teletraffic Congress*, pp. 23–30.

[15] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proc. 2011 ACM SIGCOMM*, vol. 11, pp. 242–253.

[16] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center network," in *Proc. 2011 USENIX Conference on Networked Systems Design and Implementation*, pp. 23–23.

[17] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in *NSDI*, vol. 10, pp. 19–19, 2010.

[18] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," under submission, 2013.



(a) Fat Tree K=4 (b) Clique K_8 (c) Random Topology R1 - 14 switches, 19 links

Fig. 6. Goodput (Mb/s): comparison between CG2, ECMP, SPAIN, and STP using Fat Tree, Clique K_8, and random topology.

[19] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," in *Proc. 2010 ACM International Conference*, p. 15.

[20] W. Wu, Y. Turner, and M. Schlansker, "Routing optimization for ensemble routing," in *Proc. 2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*, pp. 97–98.

[21] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., 1990.

[22] V. Chvatal, *Linear Programming*. Macmillan, 1983.

[23] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: an approach to universal topology generation," in *Proc. 2001 International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 346–353.

Prof. Assi received his B.Eng. degree from the Lebanese University, Beirut, Lebanon, in 1997 and his Ph.D. degree from the City University of New York (CUNY) in April 2003. He is currently a full professor with the Concordia Institute for Infor-mation Systems Engineering, Concordia University. Before joining Concordia University in August 2003 as an assistant professor, he was a visiting researcher with Nokia Research Center, Boston, Massachusetts, where he worked on quality of service in passive optical access networks. His research interests are in the areas of networks and network design and optimization. He received the prestigious Mina Rees Dissertation Award from CUNY in August 2002 for his research on wavelength-division multiplexing optical networks. He is on the Editorial Board of IEEE COMMUNICATIONS SURVEYS & TUTORIALS, IEEE TRANSACTIONS ON COMMUNICATIONS, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. His current research interests are in the areas of network design and optimization, network modeling and network reliability. Dr. Assi is a senior member of the IEEE.

Sara Ayoubi received her Master's degree in 2012 from the Department of Mathematics and Computer Science, Lebanese American University, Beirut, Lebanon. Her research experience was mainly focused on Web Services Security and Aspect Oriented Programming. She is currently a Ph.D. student at the Concordia Institute of Information Systems Engineering, University of Concordia, Montreal, Canada. Her current research interests are in the area of Cloud Computing, Network Virtualization, Data Center Networks Design, and Optimization.

Samir Sebbah is a research associate with CISSE Concordia University. He received his MSc. degree (2003) from the University of Paris in computer science and operations research and his Ph.D. degree (2010) from Concordia University (Montreal, Canada) in electrical and computer engineering. During Oct 2010-August 2012 he held an NSERC Visiting Fellowship at Defence R&D Canada in Ottawa. His research interests are in the areas of design and optimization of transportation networks.

Khaled Bashir Shaban received the Ph.D. de-gree in 2006 from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada. His research experience in academic and industrial institutions covers a variety of domains in intelligent systems application and design. He is currently an Assistant Professor in the Department of Computer Science and Engineering, College of Engineering, Qatar University, Doha, Qatar.