S-CORE: Scalable Communication Cost Reduction in Data Center Environments

Fung Po Tso*, Konstantinos Oikonomou†, Eleni Kavvadia†,
Gregg Hamilton* and Dimitrios P. Pezaros*
*School of Computing Science, University of Glasgow, G12 8QQ, UK
†Department of Informatics, Ionian University, 49100 Corfu, Greece

***Abstract*—Maximizing return-on-investment through efficient resource usage of Data Center (DC) infrastructures is of paramount importance for Cloud service providers. Although a number of networking architectures and routing/flow scheduling protocols have been developed to offer full bisection bandwidth over such topologies, deployment costs and machine virtualization can impact the aggregate network load and result in episodes of congestion, especially at the core. In this paper, we propose S-CORE, a scalable Virtual Machine (VM) migration strategy to dynamically allocate VMs to servers so that the overall communication footprint of the active traffic flows is minimized. We first formulate the aggregate VM communication as an optimization problem whose solution minimizes a *communication cost* function. This *optimal* yet centralized approach is of high complexity and requires global information, constituting it not *scalable* for inherently dynamic DC environments. We then define a distributed *migration policy* based on local information that adapts to dynamic traffic changes and achieves significant communication cost reduction. We evaluate S-CORE over diverse aggregate load, DC topologies, and coordination policies, and we show that it can achieve up to 90% communication cost reduction, while its locality properties can be exploited for other purposes such as energy footprint reduction.**

I. Introduction

Cloud computing is steadily emerging as a dominant processing paradigm where ICT resources are outsourced to Cloud providers that offer infrastructure, platform or software as a service. Significant research effort has recently focused on the underlying Data Center (DC) infrastructure design [1][2][3][4][5], and into resource allocation mechanisms [6][7][8][9] that can maximize network and server usage efficiency. Although DCs are built on top of commodity Internet switching/routing and transport protocols, it is clear that the over-provisioning paradigm of the Internet is not sustainable, since the significant capital outlay for companies setting up a Cloud site (including server, infrastructure, electricity and networking costs) makes return-on-investment through maximization of resource usage efficiency crucial [10]. Despite early DC topologies aiming to provide full bisectional bandwidth between any server pair, in practice, network bandwidth available to servers is often over-subscribed due to the cost associated with deploying and expanding such topologies. While Cloud computing is still at its early stages and we are yet to see a full-blown application matrix over DC topologies (currently, http(s), DFS, authentication flows dominate), the observed load already constitutes a significant fraction of the topologies' total capacity [1][11].

Resource virtualization is a powerful mechanism in that respect, since it can be exploited to map available resources to current or anticipated demand. Research into Virtual Machine (VM) placement, consolidation and migration has mainly tried to efficiently allocate server-side resources, such as CPU usage, memory usage or aggregated network I/O [12][13][14][15][16]. While server-side metrics are useful for ensuring server resources are not over-subscribed and can be used to reduce the number of servers required to be powered on, they take no account of the resulting congestion, especially over the expensive core links of the network. On the contrary, recent evidence suggests that machine virtualization has itself a significant impact on Cloud environments, causing dramatic performance and cost variations which mainly relate to networking rather than software bottlenecks [17][18].

In this paper, we define and evaluate a novel, dynamic VM allocation scheme that minimizes the overall communication cost of the DC topology while adhering to server-side resource capacity boundaries. By assigning distinct link weights at the different layers of the DC infrastructure and taking into account the amount of data traffic carried over these links, a cost function of the overall communication cost is defined. Thus, communication cost is related to the intensity of the data traffic over all link levels and the aforementioned link weights which characterize various cost-incurring aspects such as investment costs, energy consumption, use of congested/oversubscribed links, etc., depending on DC operator policy. The problem of minimizing this cost function can be solved in a centralized manner to derive the optimal VM allocation (i.e., an allocation that results in the minimum overall communication cost). However, this optimal centralized approach is of high complexity and requires global information, and is therefore not scalable for the considered environments.

We propose S-CORE under which we iteratively localize pairwise VM traffic to lower-layer links where bandwidth is not as over-subscribed as it is in the core, and interconnection switches are cheaper to upgrade [19]. S-CORE naturally exploits network locality and reduces traffic over the high-cost aggregate links. At the same time, it abstracts from the topology characteristics and can be readily extended to apply to current and future DC network architectures. In contrast to existing work that uses complex centrally-controlled optimization algorithms [15][20][21], S-CORE adopts a distributed approach based on information available locally at the VM level,

a property that makes our approach scalable and realistically implementable over large-scale DC infrastructures. In addition, we have implemented four distinct VM migration policies and evaluated their efficiency and relative cost.

Simulation results show that we can significantly reduce traffic over the high-cost links at the core of the topology that have been shown to experience congestion even when lower communication layers are under-utilized [11][17]. Our results show great promise in achieving overall communication cost reduction of as high as 90% of the optimal allocation approximated by the centralized algorithm that assumes global traffic knowledge, and is in practice prohibitively expensive to implement. Our approach differentiates considerably from existing network-aware VM migration strategies that solely focus on balancing network load [9] or on clustering VM activity on a subset of servers to reduce energy consumption [22].

This work significantly contributes towards engineering DC networks that are continuously self-optimizing to avoid both underloaded and overloaded topologies, therefore avoiding performance bottlenecks while at the same time alleviating the prohibitive cost of over-provisioning.

The remainder of this paper is structured as follows. Section II presents an overview of our migration scheme and its associated definitions. Section III describes the cost analysis of VM allocation in a DC. Section IV introduces the details of our S-CORE scheme. Section V presents an evaluation using a simulation model. Section VI discusses related work, and Section VII concludes the paper.

II. System Definition

In this section, we formalize the problem of communication cost reduction and the concepts of allocation, communication level, and link weights. Let $\mathbb{V}$ be the set of VMs in the DC hosted by the set of all servers $\mathbb{S}$, such that every VM $u \in \mathbb{V}$ and every server $\hat{x} \in \mathbb{S}$. Each VM u in the DC is unique and it is assigned a unique identifier $\mathtt{ID}_u$. Furthermore, each VM is hosted by a particular server and let $\mathcal{A}$ denote an *allocation* of VMs to servers within the DC. Let $\hat{\sigma}^{\mathcal{A}}(u)$ be the server that hosts VM u for allocation $\mathcal{A}$. Obviously, $u \in \mathbb{V}$ and $\hat{\sigma}^{\mathcal{A}}(u) \in \mathbb{S}$. Let $\mathbb{V}_u$ denote the set of VMs that exchange data with VM u.

Communication among VMs in a DC is dictated by the network topology and the switching/routing algorithms employed. Typically, architectures offering redundant paths between servers are deployed to ensure high bisection bandwidth and high availability. Without loss of generality, we consider a layered reference network architecture [10] proposed by Cisco [23], as shown in Fig. 1. However, we refrain from including topological assumptions in our formulations, and therefore minor adaptations to our algorithms would make them applicable to potentially any DC topology. The effectiveness of our proposed approach to other DC topologies is demonstrated later using simulation results. As depicted in Fig. 1, the DC connects to the Internet through a set of core routers (CR). This is the highest level of communication. One level below, access routers (AR) interconnect CRs and the underlying aggregate

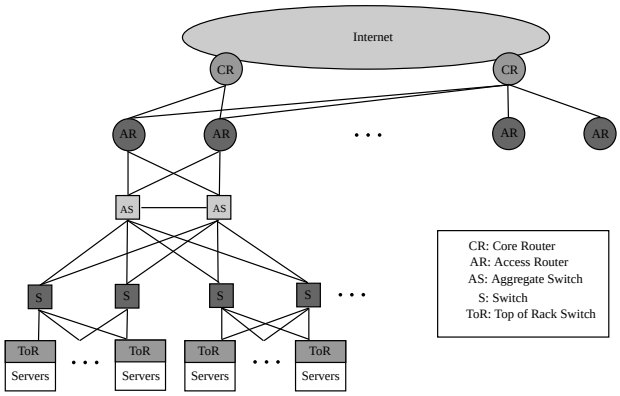


Fig. 1: The considered network architecture for data centers as proposed by Cisco [23].

switches (AS). ASes connect to switches (S) and the latter connect to switches attached at the top of each rack (ToR). Network links that connect ToR and S will be referred to hereafter as 1-*level links*, those between S and AS as 2-*level links*, etc.

Each rack is connected to the rest of the DC (and the Internet) through a ToR switch and physically hosts a number of servers (the number varies among rack types). Let $\mathtt{h}(\hat{x}, \hat{y})$ denote the *number of hops* between server $\hat{x}$ and server $\hat{y}$ along a shortest path. If a pair of servers is located at the same rack, then there is no real network link between them. In this case, the ToR switch plays a forwarding role that is a built-in capability of the rack infrastructure and therefore, $\mathtt{h}(\hat{x}, \hat{y}) = 0$. If the servers are located in different racks, then $\mathtt{h}(\hat{x}, \hat{y}) > 0$. For example, if servers $\hat{x}$ and $\hat{y}$ communicate over a switch (S), then this is a two hop communication, and $\mathtt{h}(\hat{x}, \hat{y}) = 2$, as it can be observed from Fig. 1. If they communicate over an AS, $\mathtt{h}(\hat{x}, \hat{y}) = 4$, etc.

Let $\ell^{\mathcal{A}}(u, v)$ denote the *communication level* between VM u and VM v for a given allocation $\mathcal{A}$. Obviously, if the servers hosting VMs u and v are in the same rack, then $\ell^{\mathcal{A}}(u, v) = 0$. If communication is established over 1-level links, then $\ell^{\mathcal{A}}(u, v) = 1$. In general, for the particular network topology of Fig. 1, $\ell^{\mathcal{A}}(u, v) = \frac{\mathtt{h}(\hat{\sigma}^{\mathcal{A}}(u), \hat{\sigma}^{\mathcal{A}}(v))}{2}$. Let $\ell^{\mathcal{A}}(u)$ denote the *highest communication level* for VM u for allocation $\mathcal{A}$, or $\ell^{\mathcal{A}}(u) = \max_{\forall v \in \mathbb{V}_u} \ell^{\mathcal{A}}(u, v)$.

Not all DC links are equal, and their cost depends on the particular layer they interconnect. For example, high-speed router interfaces are much more expensive than lower-level ToR switches. Therefore, in order to accommodate a large number of VMs in the DC and at the same time keep investment costs low from a provider's perspective, utilization of the "lower cost" switch links is preferable to utilization of the "more expensive" router links. The latter observation is formulated in this work by assigning a representative "weight" to every link level. In particular, let $\mathtt{c}_i$ denote the *link weight* for an i-level link per data unit (e.g., byte) representing the previously mentioned cost. Obviously, $\mathtt{c}_1 < \mathtt{c}_2 < \mathtt{c}_3 < \mathtt{c}_4$ (the higher the level, the more expensive the interface). Determining link weights, and thus the communication cost, is

based on DC operator policy and management priorities. For example, if reduction of investment costs is the priority, then link weights may be chosen to represent the market prices of the interfaces. If energy consumption, congestion, etc., is the crucial factor, then link weights can be chosen accordingly.

III. Communication Cost Analysis

The intensity of traffic exchange among each pair of VMs indicates the utilization of the intermediate links. Even though traditionally high utilization was the ultimate purpose, the objective here is to utilize (if possible) links of small link weights. Let $\lambda(u,v)$ denote the *traffic load* (or rate) in data per time unit exchanged between VM u and VM v (incoming and outgoing). Note that the traffic load is expected to vary in such highly dynamic environments. Therefore, $\lambda(u,v)$ will denote the average rate of data exchanged among VMs u and v over a certain time window, suitable to capture the dynamism of the environment.

The focus of this work is on communication levels $\ell^{\mathcal{A}}(u,v) > 0$. If $\ell^{\mathcal{A}}(u,v) = 0$, then both VMs u and v are located at the same rack, and such pairwise traffic exchange does not result in any network load that is the focus of this study. For communication level $\ell^{\mathcal{A}}(u,v) = 1$, data are exchanged over two links (i.e., $\mathtt{h}(\hat{\sigma}^{\mathcal{A}}(u), \hat{\sigma}^{\mathcal{A}}(v)) = 2$); the corresponding link weight for using each link being $\mathtt{c}_1$. For each of the links, the product $\lambda(u,v)\mathtt{c}_1$ corresponds to a (weighted) communication cost for utilizing the particular 1-level link. If the communication is through level 2 of the hierarchy (i.e., $\ell^{\mathcal{A}}(u,v) = 2$), data exchanges take place over four links, two being 2-level (weight $\mathtt{c}_2$) and two 1-level (weight $\mathtt{c}_1$) links. Eventually, the communication cost in this case corresponds to $2\lambda(u,v)(\mathtt{c}_1 + \mathtt{c}_2)$. In general, when the communication among two VMs u and v is of level $\ell^{\mathcal{A}}(u,v)$, the communication cost corresponds to $2\lambda(u,v)\sum_{i=1}^{\ell^{\mathcal{A}}(u,v)} \mathtt{c}_i$.

Given that any VM u communicates with all VMs in set $\mathbb{V}_u$, there is a *communication cost*, denoted by $C^{\mathcal{A}}(u)$, attributed to VM u, for allocation $\mathcal{A}$,

$$C^{\mathcal{A}}(u) = 2 \sum_{\forall v \in \mathbb{V}_u} \lambda(u,v) \sum_{i=1}^{\ell^{\mathcal{A}}(u,v)} \mathtt{c}_i. \tag{1}$$

It is now possible to derive an expression with respect to the *overall communication cost* for all VM communications over the DC. Let $C^{\mathcal{A}}$ denote this cost for allocation $\mathcal{A}$. Obviously, $C^{\mathcal{A}}(u)$ as $C^{\mathcal{A}} = \frac{1}{2}\sum_{\forall u \in \mathbb{V}} C^{\mathcal{A}}(u)$ ($\frac{1}{2}$ is inserted since each link is counted twice). Eventually,

$$C^{\mathcal{A}} = \sum_{\forall u \in \mathbb{V}} \sum_{\forall v \in \mathbb{V}_u} \lambda(u,v) \sum_{i=1}^{\ell^{\mathcal{A}}(u,v)} \mathtt{c}_i. \tag{2}$$

Note that Eq. (2) does not take into account traffic exchanged with applications outside the DC. For this case any shortest path is along CR, AR, AS, S, and ToR for any allocation $\mathcal{A}$.

From Eq. (2) it becomes obvious that in general, different allocations $\mathcal{A}$ correspond to different overall communication costs. The objective here is to derive a particular allocation for which the overall communication cost is minimized (i.e., optimal). Let $\mathcal{A}_{opt}$ denote an *optimal allocation*, such that $C^{\mathcal{A}_{opt}} \leq C^{\mathcal{A}}$, for any possible $\mathcal{A}$ (note that there might be more than one allocation that minimizes the overall communication cost). To the best of our knowledge this is the first time that such formulation is proposed for minimizing communication costs in a DC environment.

The objective is to derive the optimal allocation for a given DC environment and most importantly, to be able to adapt to any dynamic changes in this environment. In special cases, the optimal allocation can be easily derived. Assume, for example, the case where all active VMs can be accommodated at a single rack. In this case, it is straightforward to minimize the overall communication cost: all VMs should be hosted under the same rack. This observation is confirmed by Eq. (2), however, this is a reduced case since DCs are built to support a large number of VMs that are initially allocated in either a random or a load-balanced manner. Therefore, the normal case is that VMs are allocated in a significant fraction of – if not all – available racks.

In the general case, the optimal allocation derivation is a difficult optimization problem because of (i) its high complexity (given the number of permutations that must be considered in an exhaustive search approach) and (ii) the global knowledge required in a highly dynamic environment like a DC. Every time the traffic dynamics change, there is a need to gather that information and recompute the optimal values. Obviously, such a centralized approach does not scale with the number of VMs and the size of current DC topologies.

These observations motivate the proposal of a distributed approach under which a VM will decide whether to migrate based on information available locally, thus being scalable. Such a distributed approach, called S-CORE, is proposed and analyzed in the following section.

IV. Description and Analysis of S-CORE

The main focus in this section is on whether a VM u should migrate to some other server or not in order to achieve an overall communication cost reduction. The approach and the analysis presented in the sequel assumes that there is a *token* in the network and that the VM holding the token at a given time is the one that decides whether to migrate or not. Then, the token is passed on to another VM according to the adopted *token policy*. Different token policies are presented in the Section V.

Let *migration* of a VM u from its current location (server $\hat{\sigma}^{\mathcal{A}}(u)$) to some other server $\hat{x}$ be denoted by $u \rightarrow \hat{x}$ for allocation $\mathcal{A}$. If migration takes place, then allocation $\mathcal{A}$ changes; let $\mathcal{A}_{u\rightarrow\hat{x}}$ denote the new allocation. Assuming that migration $u \rightarrow \hat{x}$ did take place, there is a new communication cost $C^{\mathcal{A}_{u\rightarrow\hat{x}}}(u)$ corresponding to allocation $\mathcal{A}_{u\rightarrow\hat{x}}$. Let $\Delta C^{\mathcal{A}}_{u\rightarrow\hat{x}}(u) = C^{\mathcal{A}}(u) - C^{\mathcal{A}_{u\rightarrow\hat{x}}}(u)$ denote the *communication cost difference* that is attributed to migration $u \rightarrow \hat{x}$. The aim next is to determine the condition under which $\Delta C^{\mathcal{A}}_{u\rightarrow\hat{x}}(u) > 0$ is satisfied.

Lemma 1: Given migration $u \to \hat{x}$, there is a communication cost difference,

$$\Delta C^{\mathcal{A}}_{u\to\hat{x}}(u) = \sum_{\forall z\in\mathbb{V}_u} C^{\mathcal{A}}(z) - C^{\mathcal{A}_{u\to\hat{x}}}(z). \quad (3)$$

Proof: Migration $u \to \hat{x}$ affects the communication of all VMs $z \in \mathbb{V}_u$, in addition to that of VM u. The rest of the VMs (i.e., $\mathbb{V}\setminus\mathbb{V}_u\cup\{u\}$) are not affected and therefore, there is no change in their corresponding communication costs. For any $z \in \mathbb{V}_u$, the difference $C^{\mathcal{A}}(z) - C^{\mathcal{A}_{u\to\hat{x}}}(z)$ corresponds to the contribution of this particular VM z to the communication cost difference $\Delta C^{\mathcal{A}}_{u\to\hat{x}}(u)$. The lemma is proved by summing up $C^{\mathcal{A}}(z) - C^{\mathcal{A}_{u\to\hat{x}}}(z)$, $\forall z \in \mathbb{V}_u$. ∎

Lemma 2: Given migration $u \to \hat{x}$, there is a communication cost difference,

$$\Delta C^{\mathcal{A}}_{u\to\hat{x}}(u) = 2\sum_{\forall z\in\mathbb{V}_u}\lambda(z,u)\left(\sum_{i=1}^{\ell^{\mathcal{A}}(z,u)}\mathsf{c}_i - \sum_{i=1}^{\ell^{\mathcal{A}_{u\to\hat{x}}}(z,u)}\mathsf{c}_i\right). \quad (4)$$

Proof: As stated above, migration $u \to \hat{x}$ affects the communication of all VMs $z \in \mathbb{V}_u$. Given allocation $\mathcal{A}$ (before migration), then according to Eq. (1), for any $z \in \mathbb{V}_u$, $C^{\mathcal{A}}(z) = 2\sum_{\forall v\in\mathbb{V}_z}\lambda(z,v)\sum_{i=1}^{\ell^{\mathcal{A}}(z,v)}\mathsf{c}_i$, which can be written as $C^{\mathcal{A}}(z) = 2\sum_{\forall v\in\mathbb{V}_z\setminus\{u\}}\lambda(z,v)\sum_{i=1}^{\ell^{\mathcal{A}}(z,v)}\mathsf{c}_i + 2\lambda(z,u)\sum_{i=1}^{\ell^{\mathcal{A}}(z,u)}\mathsf{c}_i$.

Suppose that migration $u \to \hat{x}$ does take place. Considering the new allocation $\mathcal{A}_{u\to\hat{x}}$, the corresponding communication cost for any VM $z \in \mathbb{V}_u$ can be written as follows (similarly as before), $C^{\mathcal{A}_{u\to\hat{x}}}(z) = 2\sum_{\forall v\in\mathbb{V}_z\setminus\{u\}}\lambda(z,v)\sum_{i=1}^{\ell^{\mathcal{A}_{u\to\hat{x}}}(z,v)}\mathsf{c}_i + 2\lambda(z,u)\sum_{i=1}^{\ell^{\mathcal{A}_{u\to\hat{x}}}(z,u)}\mathsf{c}_i$.

At the same time, migration $u \to \hat{x}$ does not affect VMs $z \in \mathbb{V}_z\setminus\{u\}$. Consequently, $\ell^{\mathcal{A}}(z,v) = \ell^{\mathcal{A}_{u\to\hat{x}}}(z,v)$, $\forall z \in \mathbb{V}_z\setminus\{u\}$. Eventually, $C^{\mathcal{A}}(z) - C^{\mathcal{A}_{u\to\hat{x}}}(z) = 2\lambda(z,u)\left(\sum_{i=1}^{\ell^{\mathcal{A}}(z,u)}\mathsf{c}_i - \sum_{i=1}^{\ell^{\mathcal{A}_{u\to\hat{x}}}(z,u)}\mathsf{c}_i\right)$, for any $z \in \mathbb{V}_u$. Based on Lemma 1, by summing up Eq. (3) $\forall z \in \mathbb{V}_u$, the lemma is proved. ∎

The following lemma derives an expression with respect to the *overall communication cost difference* $C^{\mathcal{A}} - C^{\mathcal{A}_{u\to\hat{x}}}$, denoted by $\Delta C^{\mathcal{A}}_{u\to\hat{x}}$.

Lemma 3: Given a movement $u \to \hat{x}$, the overall communication cost difference is given by,

$$\Delta C^{\mathcal{A}}_{u\to\hat{x}} = 2\sum_{\forall z\in\mathbb{V}_u}\lambda(z,u)\left(\sum_{i=1}^{\ell^{\mathcal{A}}(z,u)}\mathsf{c}_i - \sum_{i=1}^{\ell^{\mathcal{A}_{u\to\hat{x}}}(z,u)}\mathsf{c}_i\right). \quad (5)$$

Proof: Given that the overall communication cost $C^{\mathcal{A}}$ can be expressed as $C^{\mathcal{A}} = \frac{1}{2}\sum_{\forall z\in\mathbb{V}} C^{\mathcal{A}}(z)$, it can also be written as, $C^{\mathcal{A}} = \frac{1}{2}\sum_{\forall z\in\mathbb{V}\setminus\mathbb{V}_u\cup\{u\}} C^{\mathcal{A}}(z) + \frac{1}{2}\sum_{\forall z\in\mathbb{V}_u} C^{\mathcal{A}}(z) + \frac{1}{2}C^{\mathcal{A}}(u)$. Similarly, when migration $u \to \hat{x}$ takes place, $C^{\mathcal{A}_{u\to\hat{x}}} = \frac{1}{2}\sum_{\forall z\in\mathbb{V}\setminus\mathbb{V}_u\cup\{u\}} C^{\mathcal{A}_{u\to\hat{x}}}(z) + \frac{1}{2}\sum_{\forall z\in\mathbb{V}_u} C^{\mathcal{A}_{u\to\hat{x}}}(z) + \frac{1}{2}C^{\mathcal{A}_{u\to\hat{x}}}(u)$.

Since migration $u \to \hat{x}$ does not affect the communication of VMs $v \in \mathbb{V}\setminus\mathbb{V}_u\cup\{u\}$, there is no change in the communication level or communication costs for these VMs, and therefore $C^{\mathcal{A}}(z) = C^{\mathcal{A}_{u\to\hat{x}}}(z)$, $\forall v \in \mathbb{V}\setminus\mathbb{V}_u\cup\{u\}$.

Consequently, $\Delta C^{\mathcal{A}}_{u\to\hat{x}}$ can be expressed by $C^{\mathcal{A}} - C^{\mathcal{A}_{u\to\hat{x}}} = \frac{1}{2}\sum_{\forall z\in\mathbb{V}_u} C^{\mathcal{A}}(z) + \frac{1}{2}C^{\mathcal{A}}(u) - \frac{1}{2}\sum_{\forall z\in\mathbb{V}_u} C^{\mathcal{A}_{u\to\hat{x}}}(z) - \frac{1}{2}C^{\mathcal{A}_{u\to\hat{x}}}(u) = \frac{1}{2}\left(\sum_{\forall z\in\mathbb{V}_u} C^{\mathcal{A}}(z) - \sum_{\forall z\in\mathbb{V}_u} C^{\mathcal{A}_{u\to\hat{x}}}(z)\right) + \frac{1}{2}\left(C^{\mathcal{A}}(u) - C^{\mathcal{A}_{u\to\hat{x}}}(u)\right)$. It is derived that $\Delta C^{\mathcal{A}}_{u\to\hat{x}} = \frac{1}{2}\left(\sum_{\forall z\in\mathbb{V}_u} C^{\mathcal{A}}(z) - C^{\mathcal{A}_{u\to\hat{x}}}(z)\right) + \frac{1}{2}\left(C^{\mathcal{A}}(u) - C^{\mathcal{A}_{u\to\hat{x}}}(u)\right)$. Based on Eq. (3), and (4), $C^{\mathcal{A}} - C^{\mathcal{A}_{u\to\hat{x}}} = 2\sum_{\forall z\in\mathbb{V}_u}\lambda(z,u)\left(\sum_{i=1}^{\ell^{\mathcal{A}}(z,u)}\mathsf{c}_i - \sum_{i=1}^{\ell^{\mathcal{A}_{u\to\hat{x}}}(z,u)}\mathsf{c}_i\right)$, and the lemma is proved. ∎

Migration of a VM from one server may be a complicated task depending on the DC type. Even if it takes place instantaneously, it requires copying of the VM environment, configuration activities (e.g., IP addresses, routing tables), etc., that correspond to a management overhead for the DC operator. Obviously, for a certain VM migration to take place, the described overhead (referred to hereafter as *migration cost* and denoted by c_m) should be compensated by the gain of the overall communication cost reduction. Therefore, condition $\Delta C^{\mathcal{A}}_{u\to\hat{x}} > c_m$ should be satisfied. The following theorem provides the fundamental condition that needs to be satisfied for a migration $u \to \hat{x}$ to take place.

Theorem 1: When migration $u \to \hat{x}$ takes place, the overall communication cost compensates for the migration cost c_m, if and only if, $\sum_{\forall z\in\mathbb{V}_u}\lambda(z,u)\left(\sum_{i=1}^{\ell^{\mathcal{A}}(z,u)}\mathsf{c}_i - \sum_{i=1}^{\ell^{\mathcal{A}_{u\to\hat{x}}}(z,u)}\mathsf{c}_i\right) > c_m$.

Proof: First, in order to allow migration $u \to \hat{x}$, $C^{\mathcal{A}} > C^{\mathcal{A}_{u\to\hat{x}}}$ should be satisfied or $C^{\mathcal{A}} - C^{\mathcal{A}_{u\to\hat{x}}} > 0$, or $\Delta C^{\mathcal{A}}_{u\to\hat{x}} > 0$. The second requirement is that the gain of the overall communication cost should compensate for the migration cost; therefore $\Delta C^{\mathcal{A}}_{u\to\hat{x}} > c_m$ should be satisfied. According to Eq. (5), the latter is satisfied if $\sum_{\forall z\in\mathbb{V}_u}\lambda(z,u)\left(\sum_{i=1}^{\ell^{\mathcal{A}}(z,u)}\mathsf{c}_i - \sum_{i=1}^{\ell^{\mathcal{A}_{u\to\hat{x}}}(z,u)}\mathsf{c}_i\right) > c_m$, is also satisfied and the theorem is proved. ∎

Based on the condition of Theorem 1, the following migration policy for virtual machines is proposed.

The S-CORE Migration Policy: A VM u migrates from server $\hat{\sigma}^{\mathcal{A}}(u)$ to another server $\hat{x}$, provided that $\sum_{\forall z\in\mathbb{V}_u}\lambda(z,u)\left(\sum_{i=1}^{\ell^{\mathcal{A}}(z,u)}\mathsf{c}_i - \sum_{i=1}^{\ell^{\mathcal{A}_{u\to\hat{x}}}(z,u)}\mathsf{c}_i\right) > c_m$, is satisfied.

Apart from the token policy that will be discussed in the following section, there are some important features of VM migration that need to be highlighted. In particular, the condition of Theorem 1 relies on information that is available locally at a given VM u. First, communication level $\ell^{\mathcal{A}}(z,u)$ for $z \in \mathbb{V}_u$ requires knowledge of the location of u and any VM $z \in \mathbb{V}_u$ exchanging data with it. This is achieved by assigning servers IP addresses from a subnet associated with each rack. A VM u can then probe the network to identify the number of hops between it and any other VM (i.e., by using `traceroute`). This is possible because many aggregate switches are actually layer 3 switches that respond to ICMP packets (as do the aggregate and core routers), and the hosting server is the first hop from a VM, providing enough topological information for a VM to accurately identify communication

levels between itself and any other VM. Link weights $\mathtt{c}_i$ and migration cost c_m are static and can be easily available locally to each VM. However, traffic load $\lambda(u, v)$ is not generally available, since traffic dynamics are expected to change in the considered environment. By considering traffic loads over a time period, the amount of data exchanged between VM u and VM v can be made available. Assuming that each VM is capable of monitoring its own incoming and outgoing traffic, then an estimation of the particular value of $\lambda(u, v)$ can be derived at VM u. Note that this approach, even though subject to the accuracy of periodical estimations, can adapt to the dynamic changes of the DC traffic depending on the size of the temporal window. For the rest of this work, it will be assumed that VMs are capable of monitoring traffic and derive accurate traffic load estimations.

From the previous discussion, it becomes obvious that S-CORE relies on information that is or can become available at each VM. According to the migration criterion, a VM may or may not migrate to some other server. If it migrates, Theorem 1 ensures that the overall communication cost will be reduced. This local decision that requires local information and eventually reduces a global cost metric (i.e., the overall communication cost), is a scalable alternative to the proposed centralized approach presented in the previous section.

In order to realize the proposed S-CORE, each VM should be able to support some simple but necessary operations. The first one is to support the traffic monitoring mechanism and be able to estimate temporal traffic load. The second is related to the VMs' ability to migrate, i.e., to move from one server to another within the DC, undergo configuration activities, etc. Finally, a mechanism for receiving and sending the token to the next VM according to the token policy is also required. The particulars of four distinct token policies employed are presented in the following section.

V. Evaluation

A. Token Policies

A first approach for a simple distributed token policy is to employ a basic round-robin mechanism. The *round-robin token policy* passes the token among VMs based on their IDs in an ascending order. In particular, starting from the VM with lowest ID, denoted as v_0, the token then passes to VM u such that $\mathtt{ID}_{v_0} < \mathtt{ID}_u < \mathtt{ID}_z$, for any $z \in \mathbb{V} \backslash \{v_0, u\}$. Let $u \leftarrow v \oplus 1$ denote that VM u is the one that follows VM v, or that there is no other VM x such that $\mathtt{ID}_u > \mathtt{ID}_x > \mathtt{ID}_v$.

From an implementation viewpoint, a token is a message formed as an array of entries, as depicted in Fig. 2, with each entry carrying a 32-bit VM ID value, capable of representing over 4 billion IDs before recycling, and an 8-bit communication level. Entries are stored in ascending order by VM ID. The size of the message is of the order of the number of VMs (i.e., $|\mathbb{V}|$) in the network. VM ID allocation is normally handled by a DC VM instance placement manager, which is part of the underlying DC network fabric. Failure recovery when the token is lost (due to a process or communication failure) can be addressed by the classic Gallager, Humblet and Spira distributed leader election algorithm [24] wherein a minimum-weight spanning tree with a single leader is constructed using only the local knowledge initially available at nodes.

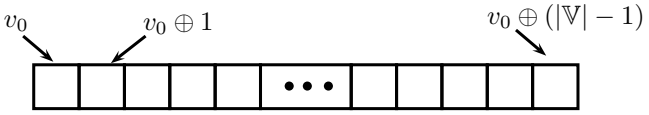


Fig. 2: Graphical representation of the token message structure.

The basic round-robin policy may not be efficient in all cases, such as when it is passed to a VM that will not migrate, wasting an iteration. A *global token policy* could eliminate this inefficiency through a centralized process that builds and distributes the token based on the largest potential communication cost reduction. Such a policy requires a token holding only the ordered VM IDs as the communication levels are centrally determined. However, maintaining global information is a costly requirement since (i) DCs are typically large-scale; and (ii) dynamic changes make information previously gathered over short timescales obsolete. Obviously, a *distributed token policy* is needed, based on locally available information.

In our distributed token policy the token passes in a round-robin manner exclusively among VMs for whom network communication passes through the highest-layer links in the network. Links are most costly at this level and it is therefore reasonable to assume that migration is likely to take place, greatly reducing the overall communication cost. The highest communication level is initialized at zero for all VMs. When the token is held by VM u then $\mathtt{l}_u$ can be updated since VM u is aware of its own highest communication level $\ell_u^{\mathcal{A}}$. VM u is also aware of the communication level of those VMs it communicates with (i.e., $v \in \mathbb{V}_u$). Therefore, it can update the corresponding entries $\mathtt{l}_v \leftarrow \ell^{\mathcal{A}}(u, v)$ accordingly. This update takes place only if the existing estimation $\mathtt{l}_v$ is smaller than the new value $\ell^{\mathcal{A}}(u, v)$. The token is passed to the next VM at this communication level, otherwise the token is passed to a VM at the next lowest level. If no VM suitable for migration is found, the policy restarts from the VM belonging to the highest communication level with the lowest ID. The details of the distributed token policy proposed here are presented in Algorithm 1.

A key feature of the distributed token policy is the communication level for VMs. A VM knows only the communication levels between itself and connected VMs. The approach followed here is based on an *estimation of the highest communication level* denoted by $\mathtt{l}_u$, the 8-bit value carried within the token message entries (see Fig. 2). The communication level is determined using `traceroute` to map the hops between two VMs, as described in Section IV.

We also present a *load-aware token policy* variant of our distributed token policy, which considers the aggregate network load (incoming and outgoing) for each VM. VMs at the same communication level, but with higher aggregate load, receive the token first, requiring only a small number of extra comparisons for the token passing decision than the distributed

Algorithm 1 Distributed Token Policy

```
1: cl ← l_u                                   ▷ cl maintains the current value of l_u
2: found ← FALSE                              ▷ Flag regarding next VM

3: for ∀v ∈ V_u do                            ▷ Update VMs connected to u
4:     if l_v < ℓ^A(u, v) then
5:         l_v ← ℓ^A(u, v)

6: z ← u ⊕ 1                                  ▷ Pick the next VM after u
7: while cl ≥ 0 && !found do
8:     while l_z ≠ cl do
9:         z ← z ⊕ 1                          ▷ Pick the following VM
10:    if l_z ← cl then
11:        found ← TRUE                       ▷ Next node is found
12:    else                                   ▷ Next node is not found at this level
13:        cl ← cl − 1                        ▷ Go to a lower level
14:        z ← v_0                            ▷ and start from the beginning

15: if !found then                            ▷ No unchecked VMs are left
16:     Pick VM z : min_ID_x{∀x ∈ V : l_x = max_∀v∈V(l_v)}
17: Send token to VM z
```

token policy.

Simulation results in Section V-D reveal the performance and communication cost reduction of each of the four algorithms discussed.

B. Experimental Setup

We have simulated the communication cost reduction VM migration algorithms on the DC topology depicted in Fig. 1 using the **ns-3** network simulator [25].

In our simulation environment, a single VM is modeled as a socket application which communicates with one or more others in the network. Similar to actual virtualization, each server has a VM hypervisor network application to manage a collective number of VMs, supporting in-migration (when one or more VMs move into a server) as well as out-migration (when one or more VMs move out of a server). The simulated topology is comprised of 2560 hosts (128 ToR switches, 20 hosts per rack). The topology can fully capture hierarchical link oversubscription at aggregate and core links. Hence, results yielded from this topology should scale to current DC network topologies, consisting of tens of thousands of servers, without loss of generality.

Each host can accommodate at most 16 VMs, to model a typical commodity DC server's capability. Assuming each server is equipped with 16GB RAM and 8 cores, 16 VMs can safely operate concurrently with 2 VMs per core and each VM occupying 1GB of RAM. Our simulated DC topology can accommodate up to $2,560 \times 16 = 40,960$ VMs.

During the simulation, each VM has 10 random outgoing connections, giving on average 20 bidirectional connections per VM. We have considered practical bandwidth limitations such that the aggregate bandwidth required by all VMs in a host does not exceed the network capacity of the physical host. Therefore, a VM migrates only when Theorem 1 is satisfied and the target host has sufficient system resources and bandwidth to accommodate it.

The four token policies discussed in Section V-A have been implemented and evaluated on our topology under varying aggregate DC load. We have defined the load level as the ratio of aggregate throughput of all VMs to the overall topology capacity, considering light (30%), medium (50%) and heavy (70%) topology load. We also simulated a scenario where the number of VMs per host and number of outgoing connections per VM are randomly generated, resulting in an aggregate traffic load of approximately 40%, halfway between the light and medium loads. We assume the link weight cost, c_i, will grow exponentially for each layer, so we set $c_1 = e^0$, $c_2 = e^1$, $c_3 = e^3$ and $c_4 = e^5$. Migration cost c_m, is set to zero for the time being to allow for a fair comparison among the centralized approach and S-CORE. However, as a DC operator may wish to associate a cost with migration, e.g., for each VM moved outwith an upper limit within a given time period, so as to limit negative effects of migration, results for various values of c_m are presented later.

C. Computation of Centralized Optimal Values

In order to benchmark the performance of S-CORE, having an optimal value based on global knowledge of the traffic dynamics is important. However, an exhaustive search across all permutations to find the optimal distribution of VMs that minimizes the overall communication cost for the topology is computationally prohibitive. For example, assuming communication happening within a rack has a cost of zero, with $16 \times 20 = 320$ VMs per rack it will need to explore at least $\binom{40,960}{320}$ combinations. We have therefore employed a more tractable heuristic search alternative, using a genetic algorithm (GA). Our GA has been implemented as part of the ns-3 library to compute approximate values for the simulation scenarios. Observing that VMs densely packed into racks minimizes the communication cost by exploiting locality, we assume that the optimal distribution(s) exists as a densely-packed VM distribution. The GA starts with a population consisting of $1,000$ individuals representing densely-packed VM distributions, each of which may or may not be an

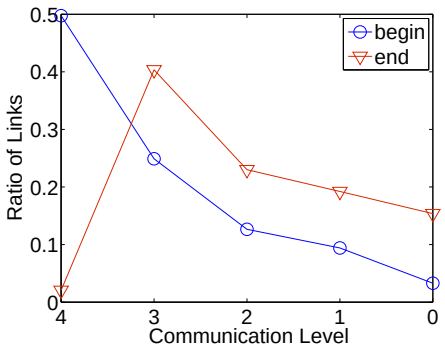


Fig. 3: Reduction of high level communication links in optimal centralized approach.

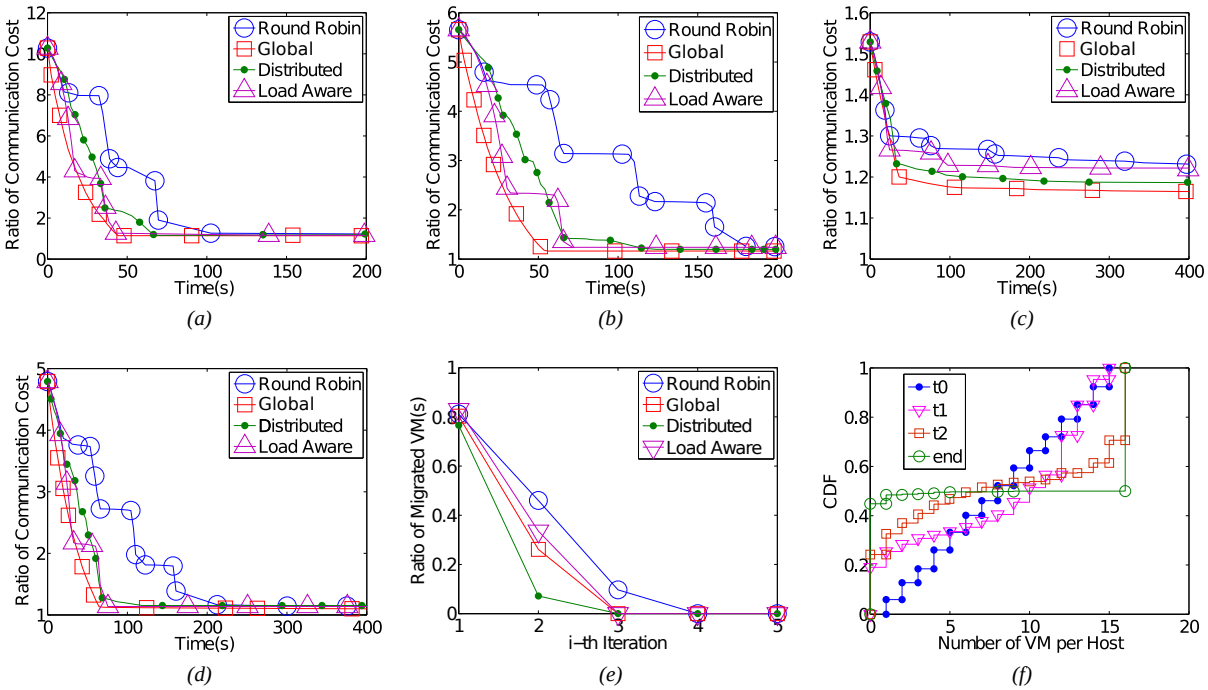


Fig. 4: Convergence time and communication cost reduction with respect to approximate values for different token policies when aggregate DC utilization is (a) 30% (light) (b) 50% (medium) (c) 70% (high) (d) approx. 42% when number of VMs per host is randomly generated (e) ratio of migrated VMs with respect to all VMs in the network after each iteration during migration and (f) CDF of no. VMs per host at different timestamps during migration for randomized scenario using the distributed token policy.

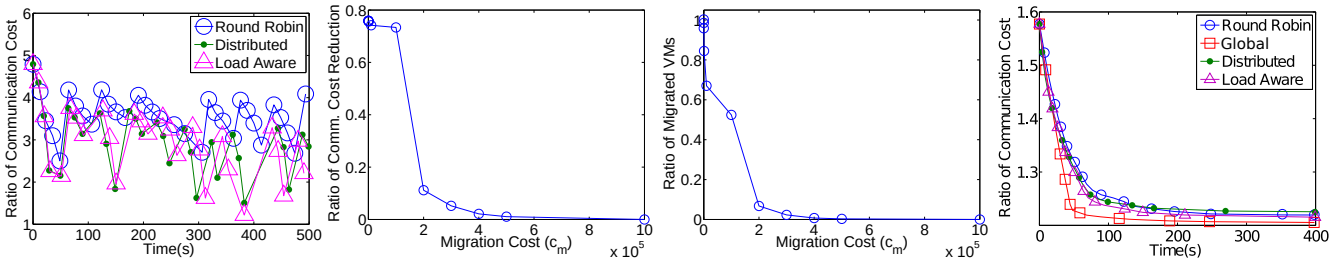
optimal solution (of VM assignments) to the problem. The crossover operator has been implemented using edge assembly crossover (EAX) and the replacement of individuals is based on tournament selection. Mutation happens by swapping a random number of VMs between racks. The GA stops when there is no significant improvement in communication cost reduction ($< 1\%$) in 10 consecutive generations. Execution time over our medium loading simulation setup is almost 12 hours using a system with 8GB RAM and a 2.66GHz quad-core CPU.

D. Experimental Results

We show in Fig. 3 the reduction of high level communication links for the centralized approach derived by the GA. This approach significantly reduces the use of the expensive, highest-level communication links from 50% to 2% of overall links. It must be noted that while the number of level 4 links is reduced, the number of level 3 (and lower) links increases. This is due to the number of randomly-instantiated connections per VM which creates a large mesh-like network, limiting further possible reduction after most communication over level 4 links has already been shifted to level 3 links and lower links.

Performance results of S-CORE are shown in Fig. 4. Figs. 4a – 4d reveal the convergence time and the ratio of communication cost reduction (with respect to optimal cost) achieved under light, medium, heavy and randomized network loading with the four different token policies. The simulation results demonstrate that S-CORE can greatly reduce the communication cost by as much as 90% of the optimal approximation in all scenarios, using only local performance information readily available at each VM. In all four scenarios, we found that the global token policy constantly exhibits best performance in terms of convergence speed and proximity to the optimal cost. However, it requires global knowledge of the traffic dynamics and can therefore be prohibitively expensive to implement in practice, even under a distributed migration algorithm. The less expensive distributed and load-aware token passing policies produce highly comparable performance to the global one. The basic round-robin policy exhibits the longest convergence time and largest difference from the approximate of the optimal amongst all four token passing policies. All token policies converge and stabilize when the VM distribution considerably reduces the overall communication cost. As it can be seen from Fig. 4a, the communication cost reduction achieved by S-CORE is as high as 90% of the approximation. Even when the aggregate load increases to 40% or 50% of the overall topology capacity, the communication cost reduction remains as high as 80% and 76%, as shown in Fig. 4b and Fig. 4d, respectively.

Fig. 4c demonstrates that while centralized and round-robin policies remain the best and worst approaches, respectively, for a heavily-loaded topology they deviate from the approximation by 18% and 26%, respectively. Interestingly, while the load aware token policy converges faster in the first 40 seconds than the distributed policy, it stabilizes with 3% less of a reduction cost. Moreover, it is shown in Fig. 4e that, when using S-CORE, the ratio of migrated VMs plummets after the second



(a) *(b)* *(c)* *(d)*

Fig. 5: (a) Random VM distribution with dynamic traffic flow (b) Ratio of communication cost reduction with the distributed token policy (c) Ratio of migrated VMs under various threshold settings with the distributed token policy and (d) Randomized scenario in Fat Tree (k=16) topology.

sequential token-passing iteration. This demonstrates that S-CORE quickly converges to a stable VM distribution within two token-passing iterations.

S-CORE aims to cut communication costs by migrating VMs to reduce the number of traffic flows routed through upper tiers in the network. Fig. 4f visualizes the change of VM distribution over time for the randomized scenario when using the distributed token policy. Initially, VMs are randomly generated and distributed over all hosts. As time passes, the CDF curve of the VM distribution rotates clockwise and becomes flatter. This implies that a significant fraction of VMs are being clustered together in certain racks. Eventually, only 56% of servers are hosting VMs, leaving 44% idle. An obvious advantage of the locality property of S-CORE is that these idle servers can be powered down to reduce the energy consumption of the DC, addressing the aims of studies on partial shutdown of servers or network elements [13][22][26].

In the above scenarios, we have assumed that pairwise VM traffic flows remain intact throughout the simulation. However, in practice, traffic flows over a DC topology are much more bursty and dynamic. To have an insightful performance indicator of how S-CORE copes with dynamic traffic, we have implemented a dynamic scenario by increasing and decreasing traffic flows to some VMs in the fully randomized setup while the migration algorithm operates. The results in Fig. 5a demonstrate that S-CORE can still adapt and converge quickly, even with dynamic traffic flows.

S-CORE is designed to work for various migration costs c_m because some migrations should not be allowed if the migration cost outweighs the gain. Clearly, the results shown in Fig. 5b and Fig. 5c illustrate that, as the weighting of c_m increases, fewer VMs are migrated. This was evaluated using the distributed token policy but it generalises to all token policies in S-CORE. Interestingly, we also found that by increasing c_m from zero to 1×10^5, the ratio of migrated VMs with respect to all VMs in the network has greatly dropped from 98% to 67% whereas the ratio of communication cost reduction just slightly deteriorates by 2%, from 78% to 76%. However, the communication cost reduction plunges sharply if we further increase c_m beyond 1×10^5. This phenomenon demonstrates that only a small fraction of VM migrations attribute significantly to a major reduction in communication cost, and having the flexibility to set c_m is important for network operators.

We have also implemented and simulated S-CORE for a fat tree topology (k=16, 1024 servers) under the randomized scenario. The fat tree topology differs from our topology, illustrated in Fig. 1, as it is a multi-rooted tree with a highest communication level of 2. Clearly, Fig. 5d illustrates that the fat tree topology exhibits significant communication cost reduction. Despite a lower saving in communication cost when compared to the legacy tree topology in Fig. 4d, it is essential to note that our low-cost S-CORE algorithm can be successfully applied to other DC topologies. We therefore believe that S-CORE can be effectively utilised in any DC network topology where there is a cost associated with communication over particular links.

VI. Related Work

Existing work has covered the areas of VM placement, migration and consolidation. VM placement tackles the problem of where to locate VM images in a DC when they are initially instantiated. Studies have addressed VM placement by solving sets of resource constraints [21], minimising the overall DC network cost matrix [8] and limiting thermal dissipation in the DC [27]. VM live migration [7] is typically employed to ensure VM performance, or some other system-side metric such as power usage, is not negatively impacted as conditions in the DC change over time, such as servers becoming overloaded [12]. Consolidation is the activity of reducing the number of servers on which VMs are hosted, often to achieve power savings [13][20], and is typically achieved via VM migration.

Further works addressing the problem of DC power reduction consider historical usage data during the consolidation processs [14] and make more effective use of the resources of consolidated servers that must remain powered on [15]. Service-level agreement (SLA) violation avoidance is also widely studied through the use of targeted VM migration [28][29], although the impact of migration itself on SLAs must be considered [13] or modeled [30]. Consolidation while meeting SLAs can be achieved by forecasting based on usage traces [14][16].

The migration techniques discussed above all make use of system-side performance metrics for migration decisions. Network-based migration studies address how to reduce the

number of network switches that must be powered on [22], meet SLAs by considering bandwidth consumption during migration [28] and balance network load by migrating VMs based on bandwidth utilisation statistics collected from switches [9]. Our work differentiates from these by addressing the problem of reducing the overall communication cost within the DC network.

VII. Conclusion

Machine virtualization is a powerful mechanism for Cloud providers to shape and control server load over their underlying DC infrastructures. However, virtualization can itself have a significant impact on the network dynamics of the topology, causing highly unpredictable traffic patterns and temporal congestion, especially over the core links [11][17]. Although the bottom layers of DC infrastructures consist of commodity server and switching equipment, network interfaces at the aggregation and core layers can be significantly more expensive and therefore much harder to upgrade. Also, congestion at the core layers affects a larger fraction of the overall DC performance.

In this paper, we have formulated a centralized approach for dynamic VM allocation in order to minimize an overall communication cost function. This centralized approach is of high complexity requiring global information, and it is therefore not scalable for the inherently dynamic DC environments. Hence we have proposed S-CORE, a distributed approach that migrates VMs from one server to another based on local information. We have evaluated S-CORE by considering four token policies and demonstrated that we can achieve up to 90% communication cost reduction when compared to the centralized approach. We have also shown that S-CORE can be equally applied to less hierarchical DC network architectures (e.g., Fat-Tree), constituting to S-CORE being a particularly suitable approach for scalable communication cost reduction in current and future DC environments.

References

[1] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," in *Proc. ACM SIGCOMM'09*, 2009, pp. 51–62.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM'08*, 2008, pp. 63–74.

[3] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "DCell: a scalable and fault-tolerant network structure for data centers," in *Proc. ACM SIGCOMM'08*, 2008, pp. 75–86.

[4] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: a high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM'09*, 2009, pp. 63–74.

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

[6] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. IEEE INFOCOM'11*, Apr. 2011, pp. 71 –75.

[7] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *USENIX NSDI'05*, 2005, pp. 273–286.

[8] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. IEEE INFOCOM'10*, Mar. 2010, pp. 1–9.

[9] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer, "Remedy: Network-aware steady state VM management for data centers," in *Proc. IFIP TC 6 Networking Conf.*, ser. LNCS, 2012, vol. 7289, pp. 190–204.

[10] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008.

[11] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC'10)*, 2010, pp. 267–280.

[12] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *USENIX NSDI'07*, 2007.

[13] A. Verma, P. Ahuja, and A. Neogi, "pMapper: power and migration cost aware application placement in virtualized systems," in *Proc. ACM/IFIP/USENIX Int. Conf. on Middleware*, 2008, pp. 243–264.

[14] S. Mehta and A. Neogi, "ReCon: A tool to recommend dynamic server consolidation in multi-cluster data centers," in *IEEE Network Operations and Management Symp. (NOMS'08)*, Apr. 2008, pp. 363–370.

[15] M. Cardosa, M. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *IFIP/IEEE Int. Symp. on Integrated Network Management (IM'09)*, June 2009, pp. 327–334.

[16] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing SLA violations," in *IFIP/IEEE Int. Symp. on Integrated Network Management (IM'07)*, May 2007, pp. 119–128.

[17] G. Wang and T. Ng, "The impact of virtualization on network performance of Amazon EC2 data center," in *Proc. IEEE INFOCOM'10*, Mar. 2010, pp. 1–9.

[18] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: comparing public cloud providers," in *Proc. ACM SIGCOMM Internet Measurement Conf. (IMC'10)*, 2010, pp. 1–14.

[19] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM'11*, 2011, pp. 266–277.

[20] G. Jung, M. Hiltunen, K. Joshi, R. Schlichting, and C. Pu, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures," in *IEEE Int. Conf. on Distributed Computing Systems (ICDCS'10)*, June 2010, pp. 62–73.

[21] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement," in *IEEE Int. Conf. on Services Computing (SCC'11)*, July 2011, pp. 72–79.

[22] V. Mann, A. Kumar, P. Dutta, and S. Kalyanaraman, "VMFlow: Leveraging VM mobility to reduce network power costs in data centers," in *Proc. IFIP TC 6 Networking Conf.*, ser. LNCS, 2011, vol. 6640, pp. 198–211.

[23] Cisco, "Data center: Load balancing data center services," 2004.

[24] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A distributed algorithm for minimum-weight spanning trees," *ACM Trans. Program. Lang. Syst.*, vol. 5, no. 1, pp. 66–77, Jan. 1983.

[25] "The ns-3 network simulator." [Online]. Available: http://www.nsnam.org/

[26] S. Khuller, J. Li, and B. Saha, "Energy efficient scheduling via partial shutdown," in *Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA'10)*, 2010, pp. 1360–1372.

[27] J. Xu and J. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *IEEE/ACM GreenCom'10*, Dec. 2010, pp. 179–188.

[28] A. Stage and T. Setzer, "Network-aware migration control and scheduling of differentiated virtual machine workloads," in *Proc. ICSE CLOUD'09*, 2009, pp. 9–14.

[29] D. Breitgand and A. Epstein, "SLA-aware placement of multi-virtual machine elastic services in compute clouds," in *IFIP/IEEE Int. Symp. on Integrated Network Management (IM'11)*, May 2011, pp. 161–168.

[30] S. Akoush, R. Sohan, A. Rice, A. Moore, and A. Hopper, "Predicting the performance of virtual machine migration," in *IEEE Int. Symp. on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS'10)*, Aug. 2010, pp. 37–46.

[31] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.

[32] C. M. Papadimitriou, *Computational complexity*. Reading, Massachusetts: Addison-Wesley, 1994.

APPENDIX

We show that the Optimal VM Allocation ($OVMA$) problem at hand does not have a polynomial time solution. $OVMA$ is not a decision problem but rather a typical optimization one, where the optimization goal is to compute whether there is a quantity $\mathcal{A}$ so that Eq. 2 is less than or equal to a target value J. We simplify the problem by considering only one communication link with cost c_1. In the sequel, we will show that $OVMA \in NP$ and then we reduce a known NP-Complete problem to $OVMA$ in polynomial time [31] (or in logarithmic space [32]).

In order to show that an optimization problem is in NP, the traditional way is to show that the following property is satisfied: for each "yes" instance there exists a "proof" or "certificate" of polynomial size, whereas "no" instances have no polynomial "certificates". $OVMA$ has this property since the certificate is an allocation $\mathcal{A}$ which is polynomial in the size of the input and it exists if and only if this allocation achieves the goal J.

The next step is to reduce a known NP-complete problem to $OVMA$. Note that a problem X is at least as hard as problem Y, if Y reduces to X, [31], [32]. We will consider the Graph Partitioning (GP) problem [31] which will be reduced to $OVMA$. For completeness, GP is stated below:

INSTANCE: Graph $G = (V, E)$, weights $w(v) \in \mathbb{Z}^+$ for each $v \in V$ and $l(e) \in \mathbb{Z}^+$ for each $e \in E$, positive integers K and J.

QUESTION: Is there a partition of V into disjoint sets $V_1, V_2, \ldots, V_m$ such that

$$\sum_{v \in V_i} w(v) \leq K$$

for $1 \leq i \leq m$ and such that if $E' \subseteq E$ is *the set of edges that have their two endpoints in two different sets* V_i, then

$$\sum_{e \in E'} l(e) \leq J \quad ?$$

In our reduction we shall use the version of GP *with vertex weight 1*, which is still NP-Complete for $K \geq 3$ (can be solved in polynomial time when $K = 2$ by matching [31]). Consider the following straightforward reduction:

- the set of VMs $\mathbb{V}$ is V, i.e., $\mathbb{V} = V = \{v_1, v_2, \ldots, v_n\}$,
- the traffic load $\lambda(u, v)$ between VMs u and v is defined as follows: $\lambda(v_i, v_j) = l(e)$, if in the undirected graph G there exists an edge e between u and v and is taken to be 0 if there is no edge between u and v in G,
- $K \in \mathbb{Z}^+$ is the rack capacity, i.e., how many virtual machines a rack may accommodate,
- the fact that the vertex weights are taken to be 1 satisfies the assumption that all VMs are equivalent in weight,
- the goal $J \in \mathbb{Z}^+$ for $OVMA$ is precisely the goal J of the GP, and
- the original question whether there is a partition of V into disjoint sets $V_1, V_2, \ldots, V_m$ now becomes the question whether there is an allocation of virtual machines to racks $r_1, r_2, \ldots, r_m$.

The above reduction is trivial and can be carried in polynomial time. Therefore, GP with vertex weight 1 reduces polynomially to $OVMA$, which completes the proof that $OVMA$ is NP-Complete.