Computer Communications 000 (2017) 1–11

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Embedding parallelizable virtual networks

Yu Liang[a], Sheng Zhang[b,*]

[a] Trend Micro CDC, PR China
[b] State Key Laboratory for Novel Software Technology, Nanjing University, 163 Xianlin Avenue Nanjing, Jiangsu 210023 PR China

ARTICLE INFO

Article history:
Received 28 July 2016
Revised 14 November 2016
Accepted 13 January 2017
Available online xxx

Keywords:
Network virtualization
Virtual network embedding
Parallelization

ABSTRACT

Network virtualization has been the focus of intense research interest for the last decade as it provides a promising approach to overcome the ossification of the Internet. An important problem in network virtualization is how to efficiently embed virtual networks with resource constraints into a substrate network (aka. virtual network embedding). Many research results have been reported regarding this problem. However, there has not been any focus on embedding parallelizable virtual networks, *i.e.*, the substrate infrastructure supports parallel computation and allows a virtual node to be mapped into multiple substrate nodes. For example, a map task can be split into several smaller map tasks by simply splitting its input data. To the best of our knowledge, this paper is the first attempt at gaining a better understanding on how parallelization benefits virtual network embedding. We identify the problem of embedding parallelizable virtual networks and propose two algorithms that capitalize parallelism from two different perspectives: proactive and lazy. Several extensions are designed to complement the proposed algorithms. Extensive simulations are conducted to evaluate the proposed algorithms and several design considerations.

1. Introduction

The Internet has been extremely successful in applications ranging from global commerce to communications, from national defense to entertainment monitoring. However, the multi-provider nature of the Internet and the end-to-end design of Internet Protocol are hard to support many newly-designed applications that run over the Internet, and thus become hurdles for its further evolution [1,2]. To overcome it, network virtualization was proposed to create and manage logical virtual networks that can better integrate with increasingly virtual application environments. Since its inception, it has received significant attentions [3,4], and it has been studied in a variety of prestigious projects, such as CABO [4], PlanetLab [5], and VINI [6].

In network virtualization environments, an infrastructure provider (InP) maintains a substrate network (SN), which is composed of physical server nodes and communication links, while a service provider (SP) purchases slices of substrate resources (e.g., CPU, bandwidth, and memory) from the InP and then creates his/her own customized virtual network (VN) to offer value-added service (e.g., Voice over IP, and content distribution) to end users. In doing so, customized network protocols can be easily deployed without requiring universal agreements between competing infrastructure holders. Besides, the decoupling of traditional ISPs leads to a layered architecture and provides great flexibility and diversity for service provision over the Internet.

One challenge in network virtualization is how to embed multiple virtual networks with resource constraints into a substrate network so as to efficiently utilize the substrate resource. This problem is also known as the Virtual Network Embedding (VNE) problem [7], and it is proven to be NP-complete [8]. A significant amount of research [9–19] has investigated techniques for the VNE problem. For example, Yu et al. [13] considered VNE with multi-path routing and dynamic virtual machine migration; Cheng et al. [15] studied topology-aware embedding; Zhang et al. [19] designed opportunistic resource sharing-based VNE; etc. However, there has not been any focus on virtual network embedding with substrate support for parallelization.

With parallelization, we can make the substrate network more supportive of embedding, in the sense that the substrate network supports parallel computation and allows a virtual node to be mapped to multiple substrate nodes. In doing so, multiple substrate nodes can parallel accomplish the computation to which the virtual node is dedicated. For example, with the advent of cyber-physical system and cloud computing, more and more data is generated everywhere over the Internet; when we use frameworks like Hadoop or Hive to run simple MapReduce jobs on this data, they

* Corresponding author.
E-mail addresses: leanne_liang@trendmicro.com.cn (Y. Liang), sheng@nju.edu.cn (S. Zhang).

http://dx.doi.org/10.1016/j.comcom.2017.01.007
0140-3664/© 2017 Elsevier B.V. All rights reserved.

are often data-parallel jobs [20]. Therefore, when we want to place a virtual node to multiple substrate nodes, we can split the input data into smaller pieces to achieve parallelization.

We find that, parallelization not only enables the substrate network to efficiently share its resources among virtual networks, but also makes virtual networks more reliable, as computation can quickly migrate to other substrate nodes in case a substrate node crashes. Since network virtualization is still in its infancy [13], we believe it is important to explore all possibilities to best serve its goals.

In this paper, we study the virtual network embedding problem with substrate support for parallelization and propose two algorithms that capitalize parallelism from two different perspectives. Several extensions are developed to complement the proposed algorithms. To the best of our knowledge, this is the first attempt that envisions substrate parallelization support in virtual network embedding. Extensive simulations are performed to evaluate the effectiveness of the proposed algorithms and design considerations.

The remainder of this paper is organized as follows. We discuss related work in Section 2. Section 3 introduces the notations and problem formulation. In Section 4, we propose two algorithms and some design tradeoffs. Before we conclude the paper in Section 6, we present simulation results in Section 5.

2. Related work

Virtual network embedding has attracted much attention from both industry and academia. Existing studies can be broadly classified into two types: VNE for Internet and VNE for datacenters. Both of them greatly inspired the design of our algorithms.

VNE for Internet. Simulated annealing was introduced to cope with the NP-completeness of the VNE problem in [9,21]. There, a candidate mapping solution is generated randomly, and the algorithms improve the candidate solution through iterative adjustments. Ricci et al. [9] considered bandwidth constraint only, and a substrate node was not allowed to be mapped to more than one virtual node. The tradeoff between embedding accuracy and running time was investigated in [21]. Zhu and Ammar [10] studied the embedding problem with the assumption of unlimited substrate resources, and focused on load balancing. With the same assumption, Lu et al. [12] attempted to minimize the embedding cost of a single virtual network with a backbone-star topology .

Subgraph isomorphism-based embedding was considered in [14]. Yu et al. [13] considered VNE with multi-path routing and dynamic virtual machine migration. Chowdhury et al. [11] added physical location constraints and employed linear programming and deterministic/randomized rounding techniques to achieve better coordination between node and link mappings. Chowdhury et al. [16] also studied the inter-domain mapping problem, i.e., mapping virtual networks between multiple substrate networks. Time-varying resource demands were considered in [19,22], where the authors proposed opportunistic resource sharing-based mapping algorithms, where substrate resources are shared among multiple virtual networks opportunistically.

Other than compute and network resources, topology is also an important factor that affects the quality of embedding [15,17,22]. Two metrics, critical index and popularity index, were introduced to differentiate between substrate nodes and links in [17]. Inspired by PageRank [23], Markov chain-based algorithms were developed in [15,22] to compute the ranking of substrate nodes, which further facilitates the resource allocation.

VN topologies can be quite large, and mapping such a large VN is not feasible for latency and complexity reasons. One option is to divide a VN into a set of smaller elementary clusters. Houidi et al. [24] focused on star-based VN decomposition. Beck et al. [25] let multiple substrate nodes calculate embeddings in parallel so as to spread the computational load from embedding across the entire substrate network. Different from them, this paper considers VNE from another orthotropic aspect: splitting a virtual node into multiple smaller ones and mapping them onto different substrate nodes. This could improve resource utilization efficiency and virtual network reliability.

VNE for Datacenters. When cloud tenants place their applications in datacenters, they usually wish that they can predict the performance of their applications. However, due to time-varying workloads of applications and resource contention in production datacenters, accurately reasoning about application performance is hard to achieve. Ballani et al. [26] proposed to use virtual networks (e.g., VC, VOC, and VDC) as better interfaces between cloud providers and tenants, allowing tenants to explicitly indicate their resource demands in a structural way. Secondnet [27] implemented VDC with source routing and rate limiting. Xie et al. [28] studied how to derive the time-varying resource demand model. Angel et al. [29] investigated the end to end performance isolation using the VDC abstraction. Stable embedding of virtual networks in multiple geo-distributed datacenters was studied in [30]. Pricing strategies that encourage tenants to proactively change their resource demands were designed in [31].

Our Contributions. Existing studies on virtual network embedding mainly put their attentions on finding coordinated node and link embedding [11], inter-domain embedding [16,30], physical node load balancing [21], better metrics for physical nodes [15,17], large-scale embedding [24,25], etc. Most of them hardly considered splitting a virtual node into multiple smaller ones, which can potentially improve physical resource utilization and increase virtual network requests acceptance ratio. We summarize the main contributions here as follows.

1) We introduce the notion of embedding parallelizable virtual networks through splitting each virtual node into multiple smaller ones, which help us better utilize physical resources.
2) We present two embedding algorithms that proactively or lazily utilize parallelization.
3) We provide three extensions to complement the proposed algorithms.

3. Parallelizable virtual networks

In this section, we introduce main notations (Section 3.1), motivate our design (Section 3.2), and give the problem formulation (Section 3.3).

3.1. Notations

CPU and bandwidth are the main constraints we consider in this paper, which is the typical case in almost all of the related literature on virtual network embedding so far. The notation system in this paper is similar to that in [11,13,19]. The principle behind these notations is that superscript “s” (or “v”) indicates a substrate (or virtual) network. Commonly, we model a substrate (or virtual) network as an undirected weighted graph; vertices represent nodes and edges represent links. Each vertex is associated with a CPU capacity/constraint, while each edge is associated with a bandwidth capacity/constraint.

Definition 1 (Substrate Network)**.** A substrate network can be represented by $G^s = (N^s, E^s, C^s, B^s)$, where N^s and E^s are the sets of substrate nodes and links, respectively. C^s is the set of CPU attributes, and B^s is the set of bandwidth attributes.

Let $RC^s(n^s)$ and $RB^s(e^s)$ be the residual CPU of substrate node n^s and the residual bandwidth of substrate link e^s, respectively. Let