Virtual Network Advance Reservation

H. Bai[1], F. Gu[2], J. Crichigno[3], S. Khan[4], N. Ghani[1], K. Shaban[5]
[1]Univ. of South Florida, [2]VMware, [3]Northern New Mexico College, [4]North Dakota State Univ., [5]Qatar University

***Abstract*—Network virtualization provides a promising means of hosting multiple client infrastructures over a physical substrate. Now one of the key concerns here is how to map virtual network requests onto physical nodes and links, and a wide range of algorithms have been developed. As applications expand, however, there is a growing need to support future scheduling of virtual network demands. Nevertheless, most advance reservation studies have only looked at point-to-point connections. Indeed, scheduling VN requests is a much more complex problem given the higher dimensionalities involved. Along these lines, this paper takes a first look at this area by presenting some tractable heuristic solutions. The findings herein provide a baseline for developing more advanced algorithms in the future.**

***Index Terms*—Virtual network, scheduling, advance reservation, cloud scheduling**

I. Introduction

Network virtualization is used to host multiple *virtual networks* (VN) over a shared substrate infrastructure. Namely, a VN is defined as a collection of VN nodes and VN links, where VN nodes typically provide storage/computing resources and VN links provide interconnection between VN nodes [1]. Overall, VN services are widely used to support cloud and storage extension applications, i.e., most notably *infrastructure as a service* (IaaS) offerings.

Now a key technical concern is how to map/embed VN requests onto the substrate networks, and various solutions have been proposed. However, as cloud-based services continue to evolve, there is a need to expand VN service flexibility. For example, advance reservation of VN demands can provide much better support for "one-time" events requiring high levels of cloud/datacenter support, e.g., sporting venues, conventions, webcasts, etc. Still, most network scheduling studies have only looked at point-to-point connections [2] and only take into account link bandwidth levels. Indeed, few studies (if any) have looked at scheduling larger virtual infrastructures with joint node/link resource requirements. Now [3] does introduce the related *virtual overlay network scheduling* (VONS) problem. However, demand end-points here are assumed to be fixed (pre-specified), and the VN node mapping/scheduling dimension is not addressed.

In the light of the above, this paper presents one of the first studies on virtual infrastructure (VN) scheduling. In particular, several VN node embedding/scheduling heuristics are developed, providing a good baseline from which to develop more advanced algorithms in the future. This paper is organized as follows. First, Section II presents a brief overview of the existing work on network virtualization and scheduling. Section III then presents a detailed problem formulation followed by some heuristic solutions. Finally, Section IV details some performance evaluation results followed by conclusions and future work directions in Section V.

II. Background

VN scheduling entails both *virtual network embedding* (VNE) and AR scheduling support. Hence this section presents some background on both areas. The overall goal of VNE is to map VN nodes and VN links over physical topologies [4]. Here each VN node requires a certain amout of computing capacity/storage resources, and similarly each VN link requires a certain amount of bandwidth to interconnect the VN nodes. Now various studies have looked at VNE design, which is an *NP*-hard problem [5]. Although some optimization-based schemes have been proposed here [6],[7], in general, solving such large combinatorial problems is quite difficult. Hence many graph-based heuristic schemes have been developed for scalable VNE. By and large, these heuristics can be classified into two key categories: separate node/link mapping (two-stage) [8] and joint node/link mapping (single-stage) [9].

On the other hand, AR scheduling reserves network resources at future intervals in time. Overall many different AR service models have been developed using static/dynamic start-times and/or durations [10]. Related scheduling algorithms have also been proposed to provision these services, using both optimization [11] and heuristic methodologies [12]. Other studies have also looked at broader AR request types. For example, [13] develops algorithms for reserving multicast connections and this work has strong applicability in on-line broadcasting. Similarly, [3] considers advance reservation of more generalized multi-point (overlay) requests with fixed node end-points, termed as the VONS problem. Moreover, [14] extends the work in [3] and solves the VONS optimization problem for a single incoming request. Results show lower blocking ratios and significantly better resource utilization versus some heuristic algorithms.

Clearly, [3] and [14] present some new directions in network services scheduling. Extending upon this, the effort herein presents a first look at the more general case of VN scheduling. Namely, virtual node mappings are no longer pre-determined and instead node placement is now part of the problem. Overall this added dimension poses further complexity, and some related solutions are now discussed.

III. Heuristic Solution

A heuristic VN scheduling algorithm is now presented. Consider the requisite notation first. The physical substrate network is modeled as an undirected graph $G_s = (V_s, E_s)$, where $V_s = \{v_s^1, v_s^2, ..., v_s^{|V_s|}\}$ is the set of substrate nodes and $E_s = \{e_s | e_s = (v_s^i, v_s^j) : v_s^i, v_s^j \in V_s\}$ is the set of substrate links. Here each substrate node $v_s \in V_s$ has a fixed amount of computing and storage resources, C, and

each substrate link $e_s \in E_s$ has a fixed amount of bandwidth capacity, B. Furthermore in order to provision VN node/link requests, time-varying capacity/resource levels are also defined for nodes and links, i.e., $rem_v(t)$ and $rem_e(t)$, respectively. Meanwhile a VN request is given by the 5-tuple, i.e., $r^n = (G_v^n, c^n, b^n, t_s^n, t_e^n)$, where n is the request index, $G_v^n = (V_v^n, E_v^n)$ is an undirected virtual network graph containing the set of VN nodes $v_v^n \in V_v^n$ and virtual links $e_v^n \in E_v^n$, c^n is the requested node resource level at each VN node ($c^n < C$), b^n is the requested bandwidth for each VN link ($b^n < B$), t_s^n is the start time, and t_e^n is the end time. Furthermore, a VN node-to-substrate mapping is denoted by $< v_v, v_s >$, i.e., VN node v_v embedded onto substrate node v_s. Note that assigning equivalent capacity to all VN nodes (VN links) does not limit generality here.

Now operators provisioning VN services will likely want to achieve high resource efficiency over their underlying substrate networks. In addition, associated revenue generation (cost reduction) concerns will also be very important. Hence two key metrics are introduced here. First a modified revenue is associated with provisioning a VN request:

$$Revenue(G_v^n) = (t_e^n - t_s^n) * (\sum_{e_v^n \in E_v^n} b^n/\mathbb{B} + \rho \sum_{v_v^n \in V_v^n} c^n/\mathbb{C}) \quad (1)$$

where ρ is the fraction of node resource revenue and $\mathbb{B}$ and $\mathbb{C}$ are two large numbers to normalize node/link resouces. Unlike [15], here the revenue is also dependent upon request durations, i.e., a request with a longer time duration but the same node/link resources gives higher revenue. Second, a similar duration-sensitive modification of the cost of accepting a VN request is defined as:

$$Cost(G_v^n) = (t_e^n - t_s^n) * (\sum_{e_s \in E_s} \mathcal{F}_{e_s}^{G_v^n}/\mathbb{B} + \pi \sum_{v_s \in V_s} \mathcal{N}_{v_s}^{G_v^n}/\mathbb{C}) \quad (2)$$

where π is the fraction of node resource cost, $\mathcal{F}_{e_s}^{G_v^n}$ is the total amount of bandwidth allocated on substrate link e_s for mapping the VN, and $\mathcal{N}_{v_s}^{G_v^n}$ is the total amount of node resources allocated at substrate node v_s for mapping the VN.

A two-stage graph-theoretic VN AR scheme is now presented, see Fig. 1. Here requests arrive and depart in continuous time and in an "on-line" manner. First, the VN nodes are sorted according to their node ranks and mapped in a sequential manner (steps 3-14, Fig. 1). Next, the virtual links between the mapped nodes are scheduled (steps 15-21, Fig. 1). The algorithm returns success if all nodes and links are scheduled (mapped). Now consider virtual node mapping, for which four different strategies are proposed:

- **Random Pick (RP)**: This scheme randomly picks a substrate node with enough free resources in the VN request interval. This approach essentially provides a baseline for comparison purposes.

- **Maximum Neighbor (MN)**: This scheme chooses the substrate node with the maximum number of available neighbours in $G_s'(V_s', E_s')$. This approach tries to map the most-connected substrate nodes first in order to increase the likelihood of successfully routing the virtual links.

1: Given incoming request $r^i = (G_v^i, c^i, b^i, t_s^i, t_e^i)$, generate temporary graph copy $G_s'(V_s', E_s') = G(V, E)$
2: Remove non-feasible nodes and links in $G_s'(V_s', E_s')$, i.e., $rem_v(t) < c^n$ and $rem_e(t) < b^n$ in $[t_s^i, t_e^i]$;
/ Loop and map all virtual nodes in the request*/*
3: Sort the virtual nodes based upon node rank
4: **for** $i = 1$ to $|V_v^i|$
5: Pick substrate node from V_s' based on selected node mapping approaches (RP, MN, MB, or MP)
6: **if** Failed
7: VN request r^i failed
8: Discard $G_s'(V_s', E_s')$ and temporary node mapping array
9: Exit loop
10: **else**
11: Remove substrate node from V_s'
12: Save node mapping $< v_v^i, v_s >$ in temporary node mapping array
13: **end if**
14: **end for**
/ Loop and map all virtual nodes in the request*/*
15: **if** All virtual nodes mapping successful
16: Assign unity weight to links in $G_s'(V_s', E_s')$
17: Run Dijkstra's shortest-path for each virtual link between mapped substrate nodes
18: **else**
19: VN request r^i failed
20: Drop r^i, Discard $G_s'(V_s', E_s')$ and node mapping array
21: **end if**
22: **if** All VN link connection routed
23: Setup successful
24: Reserve mapped node resources from node mapping array onto $G_s(V_s, E_s)$
25: Reserve routed link resources onto $G_s(V_s, E_s)$
26: **end if**

Fig. 1. Virtual Network Advance Reservation heuristic algorithm

- **Maximum Bandwidth (MB)**: This scheme chooses the substrate node with the most available bandwidth on its adjacent links. This approach tries to achieve link load-balancing when mapping virtual nodes, i.e., by avoiding heavily-loaded substrate links.

- **Maximum Product (MP)**: This scheme pursues a median between node and link loads. Namely, the product of the remaining node resources and total adjacent link bandwidth levels are calculated for each substrate node in $G_s'(V_s', E_s')$. Each virtual node is then mapped to the available substrate node with the maximum product.

Carefully note that a temporary copy of the network graph, $G_s'(V_s', E_s')$ is used to perform all node/link mapping computations, Fig. 1. This graph first removes any physical nodes or links which cannot accomodate a requested virtual node or link, i.e., $rem_v(t) < c^n$ and $rem_e(t) < b^n$ (non-feasible).

IV. Performance Analysis

The performance of the VN scheduling heuristics is analyzed using custom-developed *OPNET* $Modeler^{TM}$ simulation models. A substrate topology with 24 nodes/43 links (3.58 node degree) is tested where all nodes have 1,000 units of generic resource capacity and all physical links have 10,000 units of bandwidth. Meanwhile, VN request topology sizes are uniformly varied from 4-7 nodes with corresponding node capacities ranging uniformly from 10 to 30 units and link capacities from 100 to 1,000 units. Accordingly, $\mathbb{B}$ is set to 1,000, $\mathbb{C}$ is set as 30 (since the largest bandwidth/node resource can only be up to 1,000/30 in a single request), and both ρ and π are set to 1 in Eqs. 1 and 2. All VN requests have exponentialy-distributed holding and inter-arrival times with means μ and λ, and a modified Erlang load metric is defined to take into account varying numbers of link requests in VN demands, i.e., akin to [3]. For testing purposes, a scaled mean holding time of μ=10 seconds is used, and the mean inter-arrival times are adjusted according to load. All tests are done for 500,000 random VN requests.

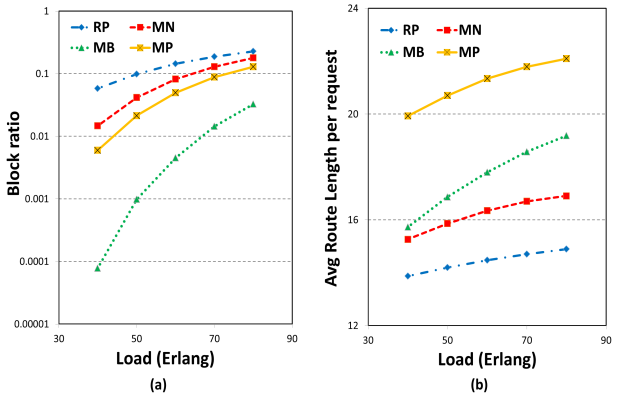

Fig. 2. VN request blocking rate (a), VN average route length (b)

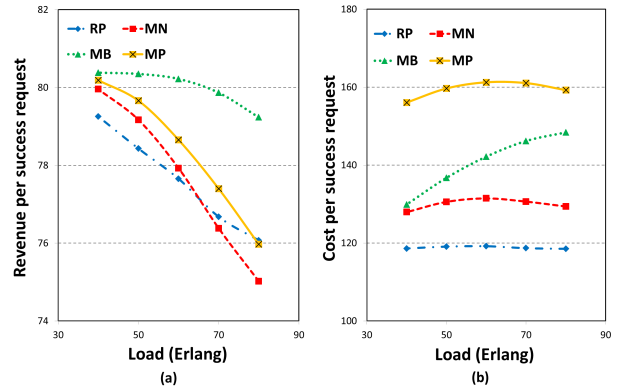

Fig. 3. Revenue per request (a), cost per request (b)

Overall blocking results are shown in Fig. 2a and indicate that the "load-aware" MB and MP schemes give the best performance, with the MB scheme doing significantly better. Meanwhile, the average path lengths of the mapped VN link connections are also shown in Fig. 2b to gauge bandwidth resource utilization. Here both load-balancing schemes (MB, MP) give better average utilization, particularly at higher load regimes. Further tests are also done to measure the revenue and cost of successfully provisioning VN requests (Fig. 3). Results here indicate that the MB scheme consistently gives the best revenue, see Fig. 3a. Meanwhile, the RP scheme gives the lowest cost for accepting a VN, whereas the MP scheme gives the highest cost. Overall, the MB scheme is deemed as the most competitive as it achieves the lowest blocking and the highest revenue, with only 10% higher cost versus the lowest-cost heuristic, i.e., RP scheme. Overall, these results clearly indicate that node mapping schemes have a huge impact on VN scheduling performance, and that the "load-aware" strategies give better overall performance.

V. Conclusions and Future Work

Growing application demands are driving the need for virtual infrastructure reservation. Hence this paper looks at the more generalized case of virtual network advance reservation and presents several heuristics. Findings indicate that different node mapping strategies have a direct influence on link load balancing, and that maximum bandwidth node selection gives the best performance. Future efforts will look at developing more advanced optimization and meta-heuristic schemes to further improve performance.

References

[1] T. Anderson, *et al*, "Overcoming the Internet Impasss Through Virtualization", *IEEE Computer Magazine*, Vol. 38, No. 4, 2005, pp. 34-41.

[2] N. Charbonneum, V. Vokkarane, "A Survey of Advance Reservation routing and Wavelength Assignment in Wavelength-Routed WDM Networks", *IEEE Communications Surveys and Tutorials*, Vol. 14, No. 4, 2012, pp. 1037-1064.

[3] F. Gu, *et al*, "Virtual Overlay Network Scheduling", *IEEE Communication Letters*, No. 8, 2011, pp. 893-895.

[4] A. Fischer, J. Botero, *et al*, "Virtual Network Embedding: A Survey", *IEEE Comm. Surveys & Tutorials*, Vol. 15, No. 4, 4^{th} Quater 2013.

[5] D. Anderson, "Theoretical Approaches to Node Assignment", unpublished manuscript, see http://www.cs.cmu.edu/dga/papers/andersenassign.ps, 2002.

[6] Z. Zhang, *et al*, "A Unified Enhanced Particle Swarm Optimization-Based Virtual Network Embedding Algorithm", *International Journal of Comm. Systems*, Vol. 26, No. 8, August 2013, pp. 1054-1073.

[7] I. Houidi, *et al*, "Virtual Network Provisioning Across Multiple Substrate Networks", *Computer Networks*, Vol. 55, No. 4, 2011, pp. 1011-1023.

[8] Y. Zhu, M. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," *IEEE INFOCOM 2006*, Barcelona, Spain, April 2006.

[9] H. Yu, *et al*, "A Cost Efficient Design of Virtual Infrastructures with Joint Node and Link Mapping", *Journal of Network and Systems Management*, Vol. 20, No. 1, 2012, pp. 97-115.

[10] N. Chowdhury, *et al*, "Virtual Network Embedding with Coordinated Node and Link Mapping", *IEEE INFOCOM 2009*, Rio De Janeiro, Brazil, April 2009.

[11] J. Zheng, *et al*, "Towards Automated Provisioning of Advanced Reservation Service in Next-Generation Optical Internet", *IEEE Communications Magazine*, Vol. 44, No. 12, 2006, pp. 68-74.

[12] J. Zheng, *et al*, "Routing and Wavelength Assignment in Optical Networks", *IEEE/ACM Transactions on Networking*, Vol. 11, No. 2, 2003, pp. 259-272.

[13] T. Entel, *et al*, "Scheduled Multicast Overlay for Bandwidth-Intensive Applications", *Proceedings of ONDM*, Essex, England, April 2012.

[14] H. Bai, *et al*, "Virtual Network Scheduling Design", *IEEE Conference on Cloud Networking (CloudNet) 2014*, Luxemborg, October 2014.

[15] M. Chowdhury, *et al*, "ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping", *IEEE/ACM Transactions on Networking*, Vol. 20, 2012, pp. 206-219.